An acceleration technique for a conjugate direction algorithm for nonlinear regression

Larry Wilmer Cornwell

1. Algorithms
I. Title

AN ACCELERATION TECHNIQUE FOR A CONJUGATE DIRECTION ALGORITHM FOR NONLINEAR REGRESSION

by

LARRY WILMER CORNWELL, 1941-

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

1972

A K Rigler 31 Jan 72

Advisor

[illegible]

O.R. Plummer

B C Gillett

M.E. Engelhardt

ABSTRACT

A linear acceleration technique, LAT, is developed which is applied to three conjugate direction algorithms: (1) Fletcher-Reeves algorithm, (2) Davidon-Fletcher-Powell algorithm and (3) Grey's Orthonormal Optimization Procedure (GOOP). Eight problems are solved by the three algorithms mentioned above and the Levenberg-Marquardt algorithm. The addition of the LAT algorithm improves the rate of convergence for the GOOP algorithm in all problems attempted and for some problems using the Fletcher-Reeves algorithm and the Davidon-Fletcher-Powell algorithm.

Using the number of operations to perform function and derivative evaluations, the algorithms mentioned above are compared. Although the GOOP algorithm is relatively unknown outside of the optics literature, it was found to be competitive with the other successful algorithms. A proof of convergence of the accelerated GOOP algorithm for nonquadratic problems is also developed.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Dr. A. K. Rigler for his aid in the selection of this thesis subject and his guidance in the preparation of this dissertation.

The author also wishes to express his thanks to the staff members of the computer centers of the University of Missouri-Rolla and of Western Illinois University. Without their cooperation, deadlines would not have been met.

Finally, the author wishes to express his thanks to his wife who endured as typist and to his family for their understanding and encouragement while preparing this dissertation.

TABLE OF CONTENTS

Table of Contents (continued) Page

Table of Contents (continued) Page

LIST OF TABLES

I. INTRODUCTION

In the study of nonlinear optimization, algorithms have been developed to optimize two types of problems:

1. Unconstrained Optimization
   Optimize $f(\vec{x})$ where f is a scalar and $\vec{x}$ is an n-vector. $f(\vec{x})$ is a linear or nonlinear function of the vector $\vec{x}$.
2. Constrained Optimization
   Optimize $f(\vec{x})$ where f is a scalar and $\vec{x}$ is an n-vector subject to the constraint $\vec{q}(\vec{x}) = 0$ (equality can be an inequality) where $\vec{q}$ is a p-vector and $n > p$. $f(\vec{x})$ and $\vec{q}(\vec{x})$ are linear or non-linear functions of the vector $\vec{x}$.

This thesis examines the unconstrained optimization problem and a particular class of algorithms, called conjugate direction algorithms, used to solve the unconstrained optimization problem. Techniques used in several successful unconstrained optimization algorithms are combined and applied to the conjugate direction algorithms. The new algorithm improves the convergence rate of Grey's Orthonormal Optimization Procedure, GOOP, and improves the convergence rate for some problems of the Fletcher-Reeves and Davidon-Fletcher-Powell algorithms.

For linear programming problems ($f(\vec{x})$ and $\vec{q}(\vec{x})$ are linear in $\vec{x}$), an algorithm, called the simplex method, has been developed which converges to the optimal solution in a finite number of steps or indicates an unbounded or infeasible solution. For nonlinear programming problems, no algorithm has been developed which will converge in a finite number of steps for all problems. Many algorithms have been developed for the nonlinear problem which are quite successful for various types of problems, but the difficulty may still exist that either the algorithm will diverge or will converge so slowly that it is useless for some problems.

One method of constrained optimization, called SUMT, is based on transforming a given constrained minimization problem into a sequence of unconstrained minimization problems. This transformation is accomplished by adding an appropriate auxiliary function of the problem constraints to define a new objective function whose minima are unconstrained in the domain of interest. By gradually removing the effect of the constraints in the auxiliary function by controlled changes in the value of a parameter, a sequence or family of unconstrained problems is generated that have solutions which converge to a solution of the original constrained problem. Because the SUMT algorithm is quite widely used for the constrained optimization problem, the need for a practical unconstrained optimization algorithm is evident.

In classifying the various unconstrained optimization algorithms for the nonlinear programming problem, one method would be: a) direct search methods, b) methods using first partial derivatives, and c) methods using second partial derivatives. The distinction between the algorithms listed above can be explained using the truncated Taylor series:

$$f(\vec{x} + \Delta\vec{x}) \simeq f(\vec{x}) + \vec{g}^T\Delta\vec{x} + \tfrac{1}{2}\Delta\vec{x}^T G \Delta\vec{x} \qquad (1.1)$$

where
$$\vec{g}^T = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right) \qquad (1.2)$$

and

$$G = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial^2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdot & & & \\ \cdot & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial^2 x_n} \end{bmatrix} . \qquad (1.3)$$

The vector $\Delta\vec{x}$ is the step needed to move from the point $\vec{x}$ to the next point $\vec{x} + \Delta\vec{x}$.

The direct search methods use only the values of the function and then make use of some other technique to determine a successful step toward the optimal point. Two well known algorithms which are classified as direct search algorithms are attributed to Hooke and Jeeves [1] and Rosenbrock [2].

By methods using first derivatives, it is meant that in the algorithm, the first partial derivatives, $\vec{g}$, are provided or approximated. Many conjugate direction algorithms fit into this classification. The GOOP algorithm, first described by Grey [3,4] and later described by Pegis, Grey, Vogl, and Rigler [5] has been proven to be a conjugate direction algorithm. The conjugate-gradient algorithm by Fletcher and Reeves [6] also fits into this classification. The variable metric algorithm due to Davidon [7] and later described by Fletcher and Powell [8] as the conjugate direction algorithm uses the first partial derivatives. Stewart's algorithm [9] is a version of Davidon's algorithm but approximates the first partial derivatives by differences. Huang [10] has shown that several of the existing conjugate-gradient algorithms and variable metric algorithms can be described in a generalized algorithm.

Methods using second partial derivatives, use the matrix of second partial derivatives, G, or use an approximation of the second partial derivatives. The Newton-Raphson algorithm is an "old method" which uses the second partial derivatives, G. The "extended Newton-Raphson" algorithm uses a one-dimensional search and has been very successful in practice.

One other possible way of classifying nonlinear programming algorithms is the form of the objective function $f(\vec{x})$. If it is necessary to write the objective function, $f(\vec{x})$, in the form of sum of squares, then

$$f(\vec{x}) = \tfrac{1}{2}\vec{h}(\vec{x})^T\vec{h}(\vec{x}) \tag{1.4}$$

where $$\vec{h}(\vec{x})^T = [h_1(\vec{x}), h_2(\vec{x}), \ldots\ldots, h_m(\vec{x})]. \tag{1.5}$$

This classification of algorithms is sometimes referred to as nonlinear least squares or nonlinear regression. Let $\vec{x}_0$ be the nominal initial value of the parameter vector and let $\Delta\vec{x}^T = (\Delta x_1, \Delta x_2, \ldots., \Delta x_n)$ be the vector of differential corrections to be found. Then if we expand the functions, $\vec{h}(\vec{x})$, in a Taylor series and truncate after the second term, we find:

$$\vec{h}(\vec{x}) \simeq \vec{h}_0 + H\Delta\vec{x} \tag{1.6}$$

where $\vec{h}_0$ is the m vector of values of the errors at $\vec{x} = \vec{x}_0$ and H is an m x n matrix of partial derivatives of the error function with respect to the parameters. Next we substitute equation (1.6) into equation (1.4):

$$f(\vec{x}) = \tfrac{1}{2}\vec{h}_0^T\vec{h}_0 + \Delta\vec{x}^TH^T\vec{h}_0 + \tfrac{1}{2}\Delta\vec{x}^TH^TH\Delta\vec{x}\ . \tag{1.7}$$

An equation for $\Delta\vec{x}$ is found by setting $\frac{\partial f}{\partial \Delta\vec{x}} = 0$ which yields

$$H^T H \Delta\vec{x} + H^T \vec{h}_o = 0 \tag{1.8}$$

or

$$\Delta\vec{x} = -(H^T H)^{-1} H^T \vec{h}_o \ . \tag{1.9}$$

This procedure has origin back to Gauss [11], but in practice the procedure did not converge. A modification proposed by Hartley [12] helped to improve the convergence rate of the Gauss procedure. Another extension of the Gauss method was suggested by Levenberg [13] and later was developed by Marquardt [14]. Other variations have recently been presented by Jones [15] and Meyer [16].

All other algorithms previously mentioned which do not require $f(\vec{x})$ to be in a form of sum of squares are still capable of solving the sum of squares problem. Thus, the nonlinear least squares or nonlinear regression algorithms are more restrictive type algorithms than those which minimize a general function, but are more effective for regression.

One purpose of this thesis is to examine the conjugate direction algorithms which have the property Q, i.e. the algorithms which will converge for a quadratic $f(\vec{x})$ in n steps, and to apply a linear acceleration technique to these algorithms to improve the rate of convergence. In this paper it is suggested that a pattern type search move be added to the conventional successful reset conjugate direction algorithms. The new algorithm consists of using a conjugate direction algorithm for n iterations, making a pattern move followed by a one-dimensional search in the pattern move direction, resetting the conjugate direction algorithm, and then repeating the cycle. Computational results are provided for several traditional test problems which indicate that the convergence rate is improved with the insertion of the linear acceleration technique.

The GOOP algorithm, mentioned above, is one algorithm which has received very little attention in current literature.* The algorithm was first developed by Grey for the design of imaging optics and later described by Pegis, Grey, Vogl, and Rigler [5] with application to filter design. Broste and Lavi [17] presented a detailed description of the algorithm, proved that the algorithm is a conjugate direction algorithm, and applied it to control problems. Although the Broste and Lavi paper made major contributions in mathematically describing the GOOP algorithm and properties of the algorithm, the numerical results recorded were misleading. In comparing algorithms, their term iteration did not have the same meaning for all methods.

A further purpose of this paper is to show that the GOOP algorithm is convergent in the nonquadratic case and to present computational results which compare the GOOP algorithm with other successful algorithms.

*Grey's method appears in production programs of several military and industrial laboratories; e.g. Aerospace Corporation, Frankfort Arsenal, White Sands Missle Range, Westinghouse Research Laboratories, and several Japanese optics companies.

II. REVIEW OF LITERATURE

Although the field of nonlinear programming is relatively new, the number of algorithms that have been suggested to solve the unconstrained optimization problem is quite large. The purpose of this review of literature is to describe those algorithms which have been successfully used in the field.

Several papers have been presented which attempt to survey the successful algorithms. Two recent articles in this category are by Powell [18] and Fletcher [19]. Papers have also been presented which attempt to unify several successful algorithms into a generalized form. Broyden [20] and Zeleznik [21] presented unified derivations of the Newton-Raphson or quasi-Newton methods. Huang [10] and Adachi [22] describe generalized variable-metric algorithms.

A. Direct Search Algorithms

The direct search algorithms are algorithms which do not calculate derivatives, but examine the objective function for directions indicating a decrease in f. The simplest direct search technique is to revise the variables of the objective function one at a time. In valley searching terminology, it has been found that this process generates estimates that fall to the solution, and most of the computer time is spent in following the valley.

Rosenbrock [2] noticed that the points in a valley were often nearly collinear and tried to identify the direction of the valley in order to use it as a search direction. Initially the variables are changed one at a time as in the process mentioned above, so that on the first iteration the initial estimate $\vec{x}_0$ is changed to $\vec{x}_0 + \lambda_1\vec{d}_1$; this estimate is changed to $\vec{x}_0 + \lambda_1\vec{d}_1 + \lambda_2\vec{d}_2$, and so on, until the complete iteration replaces the initial solution by the estimate:

$$\vec{x}_0 + \sum_{i=1}^{n} \lambda_i\vec{d}_i \ . \tag{2.1}$$

Before starting a new iteration, the set of n search directions is changed, and the first search direction is replaced by the vector:

$$\vec{d}_1^{\,*} = \sum_{i=1}^{n} \lambda_i\vec{d}_i \tag{2.2}$$

which is the change in the estimate of the solution that has been calculated. The remaining new search directions are obtained by an orthogonalization process, and then the iterative process is repeated.

The direct search algorithm developed by Hooke and Jeeves [1] also has been found to be successful. An iteration of the Hooke and Jeeves algorithm is in two parts, which are called "exploratory move" and "pattern move". The exploratory move is applied first, which is really a fine adjustment of the values of the variables. Specifically, small steps are taken along each of the coordinate directions in order to decrease the objective function; let the resultant point be, $\vec{z}$. If $f(\vec{z})$ is less than $f(\vec{x}_0)$, then $\vec{z}$ becomes the starting approximation for the next iteration, but otherwise the step is treated as a failure. An exploratory move is then made for the next variable until the n-coordinate directions are explored. The pattern move is then applied, and it changes the current estimate of the position of solution, $\vec{z}$, by the total change made in the last iteration; let the resultant point be $\vec{y}$. From this point a new exploratory move is made. The pattern moves can take long steps along valleys, while the exploratory phase can move down the side of a valley and identify its direction. The fact that $f(\vec{y})$ is permitted to be greater than $f(\vec{x}_0)$ is the special feature that causes the Hooke and Jeeves algorithm to be particularly suitable for optimizing objective functions that have curved valleys.

B. Algorithms Using First Derivatives

The literature is abundant with algorithms using first partial derivatives or approximations of the first partials. The successful algorithms which use first partials will be classified into one of the following types of algorithms: 1) steepest descent algorithms, 2) Gauss algorithms,

3) conjugate direction algorithms, and 4) rank-one algorithms.

1. Steepest Descent Algorithms

The steepest descent algorithm was first described by Cauchy [23] in 1847. If we consider a nonlinear objective function $f(\vec{x})$, the algorithm will move from the present point $\vec{x}_i$ to the next point $\vec{x}_{i+1}$ by means of the step $\Delta\vec{x}_i$, i.e.,

$$\vec{x}_{i+1} = \vec{x}_i + \Delta\vec{x}_i \ . \tag{2.3}$$

The step size $\Delta\vec{x}_i$ can be written

$$\Delta\vec{x}_{i+1} = -\alpha_i\vec{p}_i \tag{2.4}$$

where $\vec{p}_i$, an n-vector, denotes the search direction and α_i, a scalar, is the step size. Then

$$\vec{x}_{i+1} = \vec{x}_i - \alpha_i\vec{p}_i \tag{2.5}$$

and

$$f(\vec{x}_{i+1}) = f(\vec{x}_i - \alpha_i\vec{p}_i) \ . \tag{2.6}$$

If we let $\vec{p}_i$ be the gradient vector, $\vec{g}_i$, of the function f evaluated at the point $\vec{x}_i$, we determine this direction vector, $\vec{p}_i$ or $\vec{g}_i$, and perform a one-dimensional search minimizing $f(\vec{x}_i - \alpha_i\vec{g}_i)$ to find the optimum step size α_i.

It can be proven that this algorithm has the descent property, but it may do so in practice very slowly after some rapid initial progress. Slow convergence is particularly likely when the $f(\vec{x})$ contours are narrow and curved, and it happens when the path of steepest descent zigzags slowly down a narrow ridge, each iteration bringing only a slight reduction in $f(\vec{x})$. Forsythe's paper [24] presented an explaination for this type of behavior. It was shown that the iterates $\vec{x}_k$ converge to the null vector by asymptotically alternating between two directions. The even iterates are collinear vectors. The odd iterates are also collinear in another direction. Thus, the convergence of $f(\vec{x}_k)$ to the optimal value is linear and no faster than linear.

2. Gauss Algorithms

The Gauss algorithm [11] was first described in 1809. The equation (1.8) is solved for $\Delta\vec{x}$:

$$\Delta\vec{x} = -(H^T H)^{-1} H^T \vec{h}_0 \; . \tag{2.7}$$

Once the step, $\Delta\vec{x}$, is determined, then the following step is made:

$$\vec{x}_1 = \vec{x}_0 + \Delta\vec{x} \; . \tag{2.8}$$

Then the procedure is repeated using the new estimate $\vec{x}_1$ in equations (2.7) and (2.8). Unfortunately, this sequence often diverges when applied to practical problems.

To avoid convergence to a stationary point that is not a minimum, and to ensure that an iteration does not increase the value of the objective function, Hartley's Modified Gauss-Newton algorithm [12] was developed. Hartley performs the linearization process by solving the system of linear equations (2.7) to get $\vec{\Delta x}$ for a direction. Consider the function

$$f(\lambda) = f(\vec{x} + \lambda\vec{\Delta x}) \tag{2.9}$$

for $0 \le \lambda \le 1$ and let λ' be the value of λ for which $f(\lambda)$ is a minimum on the interval $0 \le \lambda \le 1$. Then let the vector

$$\vec{y} = \vec{x} + \lambda'\vec{\Delta x} \tag{2.10}$$

be the actual step taken.

One method which has been quite successfully used, which has the property of using only first derivatives, is the Levenberg-Marquardt algorithm [13,14]. Marquardt developed a compromise between the Gauss algorithm and the steepest descent algorithm. In this algorithm he defines a procedure for scaling the system of linear equations comparable with equations (2.7). We will refer to this system in the form

$$A^*\vec{\delta}^* = \vec{g}^* . \tag{2.11}$$

In the algorithm, when the iterated value, $\vec{x}_i$, becomes available, we solve the equation:

$$(A^*_i + \lambda_j I)\vec{\delta}^*_i = \vec{g}^*_i \qquad (2.12)$$

for $\vec{\delta}^*_i$. The $\vec{\delta}^*_i$ is scaled giving $\vec{\delta}_i$. Then the new trial vector is given by

$$\vec{x}_{i+1} = \vec{x}_i + \vec{\delta}_i \; . \qquad (2.13)$$

Before solving equation (2.12), a value of λ_j is selected and it must be such that

$$f(\vec{x}_{i+1}) < f(\vec{x}_i) \; . \qquad (2.14)$$

Marquardt showed that unless $f(\vec{x}_i)$ is a minimum, a sufficiently large value of λ_j can always be chosen such that the inequality (2.14) is true. Some form of trial and error is needed to find a value of λ_j which will lead to satisfaction of the inequality of (2.14).

Through use of this algorithm we always obtain, within a factor determined by λ, the maximum neighborhood in which the linearized model gives an adequate representation for our purposes. The algorithm shares with the steepest descent algorithm the ability to converge from an initial guess, $\vec{x}_0$, which may be outside the region of convergence of other methods. The algorithm also shares with the Gauss algorithm the ability to close in rapidly on the converged value of $\vec{x}$, once the vicinity of this value is reached.

Thus the algorithm combines the best features of the Gauss and steepest descent algorithms while avoiding their most serious limitations.

3. Conjugate Direction Algorithms

The conjugate direction algorithms may be characterized as algorithms which minimize the objective function for the k mod (n) iteration in the k-dimensional subspace determined by k conjugate direction vectors. These algorithms use or approximate the first partial derivatives and use strategies that would yield the exact answer if $f(\vec{x})$ were a quadratic function in n or less steps, property Q. These properties do not guarantee fast convergence when the higher derivatives of $f(\vec{x})$ are nonzero, but in practice the algorithms are extremely successful, although theoretical reasons for the success have not as yet been completely established.

The directions $\vec{p}$ and $\vec{q}$ in the space of the variables are conjugate with respect to the positive definite quadratic objective function:

$$\Phi(\vec{x}) = c + \vec{a}^T\vec{x} + \vec{x}^T G\vec{x} \tag{2.15}$$

if they are both nonzero, and if they satisfy the equation

$$\vec{p}^T G\vec{q} = 0 \cdot \tag{2.16}$$

The matrix G is the n x n matrix of second partial derivatives with respect to the $\vec{x}$-vector, $\vec{a}$ is an n-vector of first partials, and c is a scalar. The reason they are useful is that if we search in the direction $\vec{p}$, and find the point $\vec{x}_i$ that minimizes $\Phi(\vec{x}_i)$, and then we search from $\vec{x}_i$ in the conjugate direction $\vec{q}$ to reach the new estimate $\vec{x}_{i+1}$, then the new value of the objective function cannot be decreased by immediately searching again in the direction $\vec{p}$. We can now calculate the exact minimum of the quadratic function $\Phi(\vec{x})$ by the above process in n steps.

If we note that in the case where the objective function is quadratic, the condition (2.16) is the same as the equation:

$$(\vec{g}_{k+1} - \vec{g}_k)^T\vec{q} = 0 \tag{2.17}$$

where $\vec{g}_k$ is the gradient of the objective function at $\vec{x}_k$. This equation contains no explicit second derivatives so we have a means of obtaining conjugacy when only first derivatives are available. We could start with an arbitrary search direction $\vec{d}_1$ and then for k = 2, 3, ...,n, we calculate the search direction $\vec{d}_k$ to be orthogonal to the changes in the gradient vector that were caused by the moves in the directions $\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_{k-1}$.

The conjugate gradient algorithm described by Hestenes and Stiefel [25] and later developed by Fletcher and Reeves [6] use the above procedure with one addition: The first search direction, $\vec{d}_1$, is the direction of steepest descent. The later search directions, $\vec{d}_k$, are chosen such that:

$$\vec{d}_k = \vec{g}_k + \beta_{k-1}\vec{d}_{k-1} \tag{2.18}$$

where β_{k-1} is a real number. β_{k-1} can be calculated by:

$$\beta_{k-1} = \frac{\vec{g}_k{}^T\vec{g}_k}{\vec{g}^T{}_{k-1}\vec{g}_{k-1}} \quad . \tag{2.19}$$

To obtain a faster rate of convergence, Fletcher and Reeves recommend that the algorithm should be restarted with a steepest descent step after each n+1 iterations. An important advantage of the method, which is not obtained by other conjugate direction algorithms, is that it does not require storage space for any n x n matrices.

Shah, Buehler, and Kempthorne [26] described a conjugate direction method, but it had the property that for a problem with a quadratic objective function, the minimum would be found in 2n-1 or less steps. The algorithm was called PARTAN, a parallel tangents algorithm. It combined steepest descent gradient searches with acceleration moves which use certain previously determined search vectors. PARTAN attempted to capitalize on concentricity and unimodality to obtain a minimum solution.

Powell [27] described a version of a conjugate direction algorithm, but his version did not require the explicit evaluation of any derivatives.

The most widely used conjugate direction algorithm is due to Davidon [7] which he called a variable metric algorithm and later described by Fletcher and Powell [8]. To describe the Davidon algorithm, we first note that the search direction $\vec{\delta}$ of the steepest descent iteration at the point $\vec{x}$ is given by:

$$\vec{\delta} = -I\vec{g} . \tag{2.20}$$

We also note that the matrix, I, may be replaced by any positive definite matrix and the objective function will still decrease. Thus there exist some choice of this positive definite matrix which will provide the fast convergence. Therefore, the k^{th} iteration of Davidon's algorithm changes the estimate $\vec{x}_k$ to the estimate $\vec{x}_{k+1}$ by searching for the minimum of the objective function along the direction

$$\vec{\delta}_k = -H_k\vec{g}_k \tag{2.21}$$

where H_k is a positive definite matrix which is chosen with intention of enhancing the rate of convergence. To calculate H_{k+1}, the Davidon iteration adds a correction term to the matrix H_k, that depends on the two vectors:

$$\vec{\sigma}_k = \vec{x}_{k+1} - \vec{x}_k \tag{2.22}$$

and

$$\vec{y}_k = \vec{g}_{k+1} - \vec{g}_k . \tag{2.23}$$

And H_{k+1} is found by:

$$H_{k+1} = H_k - H_k\vec{y}_k\alpha_k\vec{y}_k{}^T H_k + \vec{\sigma}_k\beta_k\vec{\sigma}_k{}^T \tag{2.24}$$

where

$$\alpha_k = (\vec{y}^T{}_k H_k \vec{y}_k)^{-1} \tag{2.25}$$

and

$$\beta_k = (\vec{\sigma}^T{}_k \vec{y}_k)^{-1} . \tag{2.26}$$

The algorithm usually converges quickly.

Another conjugate direction algorithm which has been given little attention in the literature, but has been found to be successful in the field of optical design is the GOOP, Grey's Orthonormal Optimization Procedure, algorithm. Because of the lack of publicity, the algorithm is not widely used. But because the algorithm is being successfully used in several industrial and research centers, there is a need for a detailed description of the algorithm. The most current description of GOOP is by Broste and Lavi [17]. These authors provided a detailed description of the mathematical theory upon which the algorithm is based. They also perform an operations count for the algorithm. The idea of operations count is extended in this paper to include the number of operations performed for function and derivative evaluation.

The algorithm is based on solving the incremental equation (2.7). The coordinate transformation generated by a Gram-Schmidt process is used so that each of the transformed parameters can be optimized separately. Then the transformation is used as a stepwise process in which the objective function is reduced at each intermediate step. Broste and Lavi proved that the algorithm was a conjugate direction algorithm and that for an n-parameter problem the orthonormal process converges to the minimum in n steps, property Q. This version of GOOP also had the property of approximating the first partial derivatives. A more detailed description of GOOP will be provided in chapter III.

Huang [10,28] has shown that the algorithms described as conjugate gradient and variable metric can be described in a generalized algorithm. He also showed that these various algorithms can be grouped into classes of algorithms which generate the same sequence of points when given the same initial H-matrix and starting point.

Huang [28] and McCormick [29] have recommended the reset procedure for variable metric algorithms when minimizing a nonquadratic function. In the algorithms, if the minimal point of the nonquadratic function cannot be reached in n or n+1 iterations, then the algorithm can be reset:

$$H_{n+1} = H_0 \tag{2.27}$$

where H_i is the H-matrix of the ith iteration. Huang gave numerical support for the reset algorithm in his article.

4. Rank-one Algorithms

Rank-one algorithms are relatively new and have not been completely developed in the current literature. Powell [18] discusses these algorithms in his recent article. The idea is a modification of Davidon's variable metric algorithm. The procedure of calculating H_{k+1} has been changed so that the difference $(H_{k+1} - H_k)$ is a symmetric matrix of rank-one. The new formula is

$$H_{k+1} = H_k - \frac{(H_k\vec{y}_k - \vec{\sigma}_k)(H_k\vec{y}_k - \vec{\sigma}_k)^T}{(H_k\vec{y}_k - \vec{\sigma}_k)^T\vec{y}_k} \qquad (2.28)$$

where $\vec{y}_k$ and $\vec{\sigma}_k$ are defined by (2.22).

To use (2.28) in an algorithm, it is necessary to fix rules for calculating $\vec{x}_{k+1}$ from $\vec{x}_k$ and use the H_k matrix. The reason the idea is so valuable is that there are very many choices of rules such that n applications of (2.28) causes H to equal $-G^{-1}$ when the objective function is quadratic, and, because of the form of the linearization iteration, this property can lead to fast convergence.

Both Broyden [30] and Davidon proceed to define rules to calculate $\vec{x}_k$. They use:

$$\vec{x}_{k+1} = \vec{x}_k - \alpha_k H_k \vec{g}_k \qquad (2.29)$$

where α_k is a parameter. Broyden proved that, if the choice of α_k is arbitrary, except that it must not cause H_{k+1} to be singular or not positive semi-definite, and if $f(\vec{x})$ is quadratic, then H_n will equal G^{-1}. The important feature of this theorem (and the rank-one algorithms) is that it does not depend on calculating α_k by applying a one-dimensional search to minimize the objective function. Goldfarb [31] discusses sufficient conditions for the convergence of a rank-one algorithm.

C. Newton-Raphson Algorithms

The Newton-Raphson procedure estimates the position of the minimum of $f(\vec{x})$ from second and lower order terms in the Taylor series, i.e.,

$$f(\vec{x} + \Delta\vec{x}) = f(\vec{x}) + \Delta\vec{x}^T\vec{g} + \Delta\vec{x}^T G\Delta\vec{x} \tag{2.30}$$

where $\vec{g}$ and G are defined by equations (1.2) and (1.3), respectively. At the minimum of a differentiable function, $\vec{g} = 0$, so, if equation (2.30) is exact, the point $\vec{x} + \Delta\vec{x}$ is the required minimal point only if

$$\vec{g} + G\Delta\vec{x} = 0 \tag{2.31}$$

is satisfied. Since this equation is linear in $\Delta\vec{x}$, it is straightforward to calculate

$$\Delta\vec{x} = -G^{-1}\vec{g} \; . \tag{2.32}$$

If the second derivative matrix, G, is positive definite at the solution, then the iterations have quadratic convergence, provided that the initial estimate is sufficiently close to the minimal point. By quadratic convergence, second order convergence is meant and is not to be confused with property Q. If the initial estimate is poor, the Newton-Raphson algorithm may fail to converge. Also the algorithm may converge to a stationary point instead of the minimum.

To avoid convergence to a stationary point that is not a minimum, and to ensure that an iteration does not increase the value of the objective function, a one-dimensional search is performed in the direction, $\Delta\vec{x}$, of equation (2.32). Specifically, consider the function

$$f(\lambda) = f(\vec{x} + \lambda\Delta\vec{x}) \tag{2.33}$$

for $0 \leq \lambda \leq 1$ and let λ' be the value of λ for which $f(\lambda)$ is a minimum. Then let the vector

$$\vec{z} = \vec{x} + \lambda'\Delta\vec{x} \tag{2.34}$$

be the next point.

Another extension to the Newton-Raphson algorithm was suggested by Levenberg [13]. A nonnegative parameter, α, that interpolates between the steepest descent iteration and the Newton-Raphson iteration (2.32) is introduced. Then the new iteration is the replacement of the estimate,

$\vec{x}$, by the estimate, $\vec{x} + \Delta\vec{x}$, where the correction is defined by the equation

$$\Delta\vec{x} = (\alpha I - G)^{-1}\vec{g} \; . \tag{2.35}$$

This technique is often called "Damped Least Squares". In the case $\alpha = 0$, equation (2.35) reduces to the Newton-Raphson iteration while if α becomes very large the correction to $\vec{x}$ tends to have the direction of the gradient $\vec{g}$. Another version of the use of the damping factor is discussed by Buchele [32].

The most serious disadvantage of the Newton-Raphson algorithm and its extension is that they require the second derivatives of the objective function. Often it happens that the second derivatives are not available or that the user prefers not to calculate them. This problem motives the development of algorithms using only first derivatives.

Another variation of the Newton-Raphson procedure, called spiral, has been developed by Jones [15]. The basic idea is that a reduced sum of squares can always be found in the plane defined by the Newton-Raphson point and the line of steepest descent at the base point. The strategy of spiral is to search along a spiral line which starts in the direction of the steepest descent direction and then arcs back toward the Newton-Raphson point in this plane. This new algorithm has been quite successful for problems with narrow-curved valleys.

III. A LINEAR ACCELERATION TECHNIQUE FOR RESET CONJUGATE DIRECTION ALGORITHMS

The linear acceleration technique, LAT, is a numerical technique designed to improve the rate of convergence of the reset conjugate direction algorithms. This chapter describes this numerical technique as a new algorithm. The new algorithm is a combination of the reset conjugate direction algorithms, a pattern move of Hooke and Jeeves' direct search algorithm, and a linear search similar to the search performed in the Modified-Gauss-Newton algorithm described by Hartley. The LAT algorithm was found to be quite successful when GOOP (to be described in detail in chapter IV) was used as the conjugate direction algorithm. The Fletcher-Reeves and Davidon-Fletcher-Powell conjugate direction algorithms were also used and found to improve convergence for particular problems.

A. Pattern Move

In developing the direct search algorithm, Hooke and Jeeves refer to pattern moves. From a particular base point, $\vec{b}_1$, exploratory moves are made in each of the n coordinates until a decrease in the function is found. Once the decrease is found, a new base point $\vec{b}_2$, is established. The vector determined by the two base points

$$\vec{v} = \vec{b}_2 - \vec{b}_1 \tag{3.1}$$

gives the direction of the pattern move. The new point after a crude pattern move is given by

$$\vec{p} = \vec{b}_2 + \vec{v} = 2\vec{b}_2 - \vec{b}_1 \ . \qquad (3.2)$$

In their development of the direct search algorithm, Hooke and Jeeves found that the pattern move was a successful computational technique, which provided the ability to follow a valley.

In dealing with the conjugate direction algorithms, the n iterations could be considered as the exploratory moves. In working with nonquadratic functions, it is known that the conjugate direction algorithms approximate the quadratic problem. Therefore, since n iterations approximate the quadratic, a pattern move could be made after these iterations. Then if the pattern move is successful, the rate of convergence toward the minimum has been increased at very little computational expense. Since the reset conjugate direction algorithms reset at this point of their algorithm anyway, the process of minimizing the function in the pattern move direction before resetting appears to be quite feasible. If the acceleration is done at the n^{th} iteration, the work of the $n + 1^{th}$ exploration is transferred to the pattern move at no extra computational expense.

B. One-Dimensional Search

When a pattern move has been successful, it is most likely that the step length is not optimal in the pattern move direction. Thus it appears feasible that a one-dimensional search could be used to determine the optimal step in the pattern move direction. That is, determine α which will minimize

$$f(\vec{b}_2 + \alpha\vec{v}) \tag{3.3}$$

where $\vec{b}_2$ is the new base point and $\vec{v}$ is given by equation (3.1).

Since a one-dimensional search is used in most conjugate direction algorithms, no additional programming is necessarily needed. Therefore, it appears to be quite sensible to gain as much progress toward the minimal point as is possible before resetting the conjugate direction algorithm.

C. The Linear Acceleration Technique (LAT)

Combining the various segments discussed in the previous sections, the new algorithm has the following steps:

a) Select a nominal point $\vec{x}_0$.

b) Perform the conjugate direction algorithm for n iterations, moving from base point, $\vec{b}_1$, to base point $\vec{b}_2$.

c) Determine the direction of the pattern move by the equation:

$$\vec{v} = \vec{b}_2 - \vec{b}_1 . \qquad (3.4)$$

d) Find the optimal step size α', such that

$$f(\alpha) = f(\vec{b}_2 + \alpha\vec{v}) \qquad (3.5)$$

is minimized as a function of α.

e) Make the pattern move

$$\vec{x} = \vec{b}_2 + \alpha'\vec{v} \qquad (3.6)$$

where α' is the optimal α in step d). The $\vec{x}$ vector then becomes the new nominal point $\vec{x}_0$.

f) Reset the conjugate direction algorithm and return to step b).

IV. GREY'S ALGORITHM

As was indicated earlier, the GOOP algorithm is a conjugate direction algorithm which has received very little attention in the literature, but is an algorithm which is being successfully used in applied fields. Broste and Lavi [17] have proven that the GOOP algorithm is a conjugate direction algorithm and that it has property Q. However, no one has presented a formal proof of convergence for the nonquadratic problem. The convergence proofs for other successful algorithms are described by W. I. Zangwill [32], and in this chapter Zangwill's mathematical framework is used to prove the convergence of the GOOP algorithm. To prove convergence, it has been assumed that the linear acceleration technique described in chapter III is applied to the GOOP algorithm. Before presenting the proof, section A supplies a detailed description of the GOOP algorithm.

A. Description of GOOP

The GOOP process seeks to find changes $\Delta\vec{x}$ in a nominal parameter vector to reduce the criterion function

$J = \frac{1}{2}f(\vec{x})^T f(\vec{x})$. The distinction of the GOOP algorithm is the manner in solving the incremental equation

$$F^T F \Delta\vec{x} + F^T \vec{F}_0 = 0 \qquad (4.1)$$

where $\Delta\vec{x}$ is the vector of increments, $\vec{F}_0$ is an m-vector and F is m by m matrix of partial derivatives evaluated at the present point.

The GOOP algorithm generates a coordinate transformation from the $\Delta\vec{x}$-space to a $\Delta\vec{y}$-space in which the transformed version of (4.1) has a simple solution. The $\Delta\vec{y}$ obtained is transformed back to obtain $\Delta\vec{x}$ which reduces J.

The transformation uses a Gram-Schmidt orthonormalization of the linearly independent vectors given by the column of the matrix F. Let $\vec{F}_i = \partial f/\partial x_i$ be the ith column of F and let columns $\vec{G}_i$ of a new matrix G be the orthonormal vectors resulting from the orthonormalization of F. Then from the Gram-Schmidt process, the relation between F and G is given by

$$F = GB \qquad (4.2)$$

where B is an upper triangular matrix generated along with G by the orthonormalization process. Then substitute equation (4.2) into equation (4.1) which yields:

$$B^T G^T G B \Delta\vec{x} + B^T G^T \vec{F}_0 = 0 \ . \qquad (4.3)$$

Because the columns of G are orthonormal, equation (4.3) simplifies to

$$B\Delta\vec{x} + G^T\vec{F}_0 = 0 \,. \tag{4.4}$$

By defining $B\Delta\vec{x}$ to be the new variable in the $\Delta\vec{y}$-space, we can reduce equation (4.1) to

$$\Delta\vec{y} = -G^T\vec{F}_0 \,. \tag{4.5}$$

Since B is an upper triangular matrix it is easily inverted and the transformation from the $\Delta\vec{y}$-space to the $\Delta\vec{x}$-space is given by

$$\Delta\vec{x} = B^{-1}\Delta\vec{y} = -B^{-1}G^T\vec{F}_0 \,. \tag{4.6}$$

The transformation to the $\Delta\vec{y}$-space decouples the effect of the parameters in the $\Delta\vec{y}$-space on the various quadratic elements of J. In this new space each component of $\Delta\vec{y}$ can be independently adjusted to reduce J without undoing the reduction achieved by adjusting any other $\Delta\vec{y}$-component. This is so because the partials of f with respect to each $\Delta\vec{y}$-component are the orthonormal columns of G.

The Gram-Schmidt orthonormalization process is described by the equations (4.7)-(4.9):

$$\vec{G}_i = (\vec{F}_i - \sum_{j=1}^{i-1} b_{ji}\vec{G}_j)/b_{ii} \tag{4.7}$$

$$i = 1,2,\ldots.,n$$

where $$b_{ji} = \vec{G}_j^T \vec{F}_i \tag{4.8}$$

$$b_{ii} = \left|\left|\vec{F}_i - \sum_{j=1}^{i-1} b_{ji}\vec{G}_j\right|\right| \tag{4.9}$$

are the elements of the upper triangular matrix B.

An important feature of this transformation is that the solution given by equation (4.5) or equation (4.1) can be computed by a step-wise procedure. Therefore, only one component Δy_i of $\Delta\vec{y}$ is computed at each step. Then from equation (4.5), the i^{th} component of $\Delta\vec{y}$ is given by

$$\Delta y_i = -\vec{G}_i^T \vec{F}_o \ . \tag{4.10}$$

A single $\vec{G}_i$ can be computed at each step by equations (4.7)-(4.9) using previously computed orthonormal vectors and a single column $\vec{F}_i$. Therefore, one component of $\Delta\vec{y}$ can be computed at each step. If the column $\vec{C}_i$ of $C = B^{-1}$ could be computed at each step, the $\Delta\vec{x}$ corresponding to Δy_i would be given by

$$\Delta\vec{x} = \vec{C}_i \Delta y_i \ . \tag{4.11}$$

Since the orthonormalization builds the B matrix one column at each step, this process can be described as the construction of a product of matrices each of the form

$$B_i = \begin{bmatrix} 1 & & & b_{1i} & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & & b_{2i} & & & & \\ & & \cdot & \cdot & & & & \\ & & & b_{ii} & & & & \\ & & & & \cdot & & & \\ & & & & & \cdot & & \\ & & & & & & \cdot & \\ 0 & \cdot & \cdot & & & & & 1 \end{bmatrix} . \qquad (4.12)$$

Let B(k) be the matrix product

$$B(i) = B_i B_{i-1} \cdot \ \cdot \ \cdot \ B_1 \qquad (4.13)$$

then $$B = B(n) . \qquad (4.14)$$

Since B_i has a simple inverse given by

$$B_i^{-1} = \begin{bmatrix} 1 & & & -b_{1i}/b_{ii} & \cdot & \cdot & 0 \\ & \cdot & & -b_{2i}/b_{ii} & & & \cdot \\ & & \cdot & \cdot & & & \\ & & & 1/b_{ii} & & & \\ & & & & \cdot & & \\ & & & & & \cdot & \\ 0 & \cdot & \cdot & & & & 1 \end{bmatrix} , \qquad (4.15)$$

the inverse of B(i) is:

$$B(i)^{-1} = B_1^{-1} B_2^{-1} \ . \ . \ . \ B_i^{-1} . \qquad (4.16)$$

Then $$B^{-1} = B(n)^{-1} = B_1^{-1} B_2^{-1} \ . \ . \ . \ B_n^{-1} . \qquad (4.17)$$

If $B(i)^{-1}$ is computed at each step, only the elements of the ith column of B and the matrix $B(i)^{-1}$ are needed. Since in succeeding steps the ith column of $B(i)^{-1}$ is not altered, it must be equal to $\vec{C}_i$, the ith column of B^{-1}. Thus $\vec{C}_i$ can be computed at each step and is given by

$$C_{ii} = 1/b_{ii} \tag{4.18}$$

$$C_{ji} = -\sum_{k=j}^{i-1} C_{jk}b_{ki}/b_{ii} \qquad \text{if } 1 \leq j < i \tag{4.19}$$

$$C_{ji} = 0 \qquad \text{if } i < j \leq n$$

where C_{ji} is the jth component of $\vec{C}_i$. The use of equations (4.18) and (4.19) at each step eliminates the process of an explicit matrix inversion.

Thus at each step the change in $\Delta\vec{x}_i$ corresponding to $\Delta\vec{y}_i$ can be computed. Summing $\Delta\vec{x}_i$ over all steps yields

$$\sum_{i=1}^{n} \Delta\vec{x}_i = \sum_{i=1}^{n} \vec{C}_i\Delta\vec{y}_i = \sum_{i=1}^{n} -\vec{C}_i\vec{G}_i{}^T\vec{F}_0 \tag{4.20}$$

$$= -B^{-1}G^T\vec{F}_0 \quad .$$

Thus a stepwise process exists which converges in n steps for the quadratic function to the same solution as given by equation (4.6).

Let $\vec{x}_i$ be the estimate of the parameter vector after the ith step such that

$$\vec{x}_i = \vec{x}_0 + \sum_{j=1}^{i} \vec{x}_j \qquad (4.21)$$

where $\vec{x}_0$ is the initial vector.

The stepwise process of the GOOP algorithm is:

Initial Step: Set $\vec{x}_0$ and $\vec{F}_0 = f(\vec{x}_0)$.

ith Step: Compute

a) $\vec{F}_i$ = ith column of F = $\partial f/\partial x_i$

b) $\vec{G}_i$ from (4.7), (4.8), and (4.9)

c) $\vec{C}_i$ from (4.19) and (4.20)

d) $\Delta y_i = -\vec{G}_i^{T}\vec{F}_0$

e) $\vec{x}_i = \vec{x}_{i-1} + \Delta y_i \vec{C}_i$

In examining the steps above, the vector $\vec{C}_i$ represents the direction in the $\Delta\vec{x}$-space in which the change $\Delta\vec{x}_i$ is made. Since

$$BB^{-1} = B^{-1}B = I \,, \qquad (4.22)$$

then

$$B\vec{C}_i = \vec{e}_i = (0,0,\ldots,1,\ldots,0)^T \,. \qquad (4.23)$$

Therefore

$$\begin{aligned}\vec{p}_i^{\,T}(F^TF)\vec{p}_j &= \vec{C}_i F^T F\vec{C}_j \\ &= \vec{C}_i^{\,T} B^T G^T G B \vec{C}_j \\ &= \vec{C}_i B^T B \vec{C}_j \\ &= 0 \qquad \text{for } i \neq j\end{aligned} \tag{4.24}$$

which proves that the directions in GOOP are F^TF conjugate. Even though the directions generated by the conjugate gradient algorithm applied to equation (4.1) are also F^TF-conjugate, they are not in general the same directions generated by the GOOP algorithm. In examining the first direction of each method, the conjugate gradient algorithm's first move is in the direction of the negative gradient of equation (4.1). Huang's paper [28] classifies several familiar algorithms such as Fletcher-Reeves, Davidon-Fletcher-Powell, and others into a generalized algorithm. If the initial H matrix is the identity matrix, then the first move is in the direction of the negative gradient for all these algorithms. In the GOOP algorithm, the first move is in the direction of the x_1-axis or any preassigned coordinate axis. Also the directions in the GOOP algorithm are not a function of a residual and if residuals are computed, they are not orthogonal as in the conjugate gradient algorithm. In any case, both algorithms solve the quadratic programming problem in n or less iterations.

One additional feature of the GOOP algorithm is a procedure which allows the algorithm to continue seeking reductions in J without completely re-initializing the computation at an intermediate point. The vector of partials, $\vec{F}_i$, used at the ith step are calculated at the most recent estimate of $\vec{x}$ produced in the preceding step. The transformation as represented by the matrices G and B^{-1} can also be adjusted after each step in order to maintain orthonormality. If the derivatives are not supplied, f_i, given by the change in $f(\vec{x})$ at the ith step, can be used to recompute the vector of partials $\partial f/\partial y_i$ by setting

$$\partial f^*/\partial y_i = \partial f_i/\partial y_i = \vec{G}_i^* . \tag{4.25}$$

The * is to denote that $\vec{G}_i^*$ may no longer be orthonormal to $\vec{G}_j$, J = 1, 2, ... , i-1. The orthonormality can be restored by adjusting G as follows

$$\vec{G}_i = (\vec{G}_i^* - \sum_{j=1}^{i-1} g_{ji}\vec{G}_j)/g_{ii} \tag{4.26}$$

where

$$g_{ji} = \vec{G}_i^*\vec{G}_j \tag{4.27}$$

$$g_{ii} = ||\vec{G}_i^* - \sum_{j=1}^{i-1} g_{ji}\vec{G}_j|| . \tag{4.28}$$

A corresponding adjustment is made in B^{-1} by postmultiplying by

$$g^{-1} = \begin{bmatrix} 1 & & -g_{1i}/g_{ii} & & 0 \\ & \cdot & -g_{2i}/g_{ii} & & \cdot \\ & & \cdot & & \\ & & 1/g_{ii} & & \cdot \\ & & & \cdot & \cdot \\ 0 & \cdot & \cdot & & 1 \end{bmatrix} . \tag{4.29}$$

B. Convergence Of The GOOP-LAT Algorithm

The following convergence proof is directed at the new algorithm, LAT, described in chapter III where the conjugate direction algorithm is GOOP, described in section A. The definitions, lemmas, theorems, and corollaries in the Mathematical Preliminaries are taken from Zangwill [33].

1. Mathematical Preliminaries

DEFINITION 1. By point-to-set map, it is meant that for any point $\vec{z} \in V$, $A(\vec{z})$ is a set in V, i.e., $A : V \to V$.*

DEFINITION 2. An algorithm is an iterative process consisting of a sequence of point-to-set maps $A_k : V \to V$. Given a point $\vec{z}^1$, a sequence of points $\{\vec{z}^k\}_1^\infty$, is generated recursively by use of the recursion

$$\vec{z}^{k+1} \in A_k(\vec{z}^k) \tag{4.30}$$

*This notation is used by Zangwill for a point-to-set map. A more common form of notation is $A : V \to 2^V$.

where any point in the set $A_k(\vec{z}^k)$ is a possible successor point $\vec{z}^{k+1}$.

DEFINITION 3. A solution set is the set of all optimal points and a solution point is an point in Ω, the solution set.

CONVERGENCE THEOREM A. Let the point-to-set map $A : V \to V$ determine an algorithm that given a point $\vec{z}^1 \in V$ generates the sequence $\{\vec{z}^k\}_1^\infty$. Also let a solution set $\Omega \subset V$ be given.

Suppose

(1) All points $\vec{z}^k$ are in a compact set $X \subset V$.

(2) There is a continuous function $Z: V \to E^1$ such that

(a) if $\vec{z}$ is not a solution, then for any $\vec{y} \in A(\vec{z})$

$$Z(\vec{y}) < Z(\vec{z}) , \qquad (4.31)$$

(b) if $\vec{z}$ is a solution, then either the algorithm terminates or for any $\vec{y} \in A(\vec{z})$

$$Z(\vec{y}) \leq Z(\vec{z}) , \qquad (4.32)$$

and

(3) The map A is closed at $\vec{z}$ if $\vec{z}$ is not a solution.

Then either the algorithm stops at a solution, or the limit of any convergent subsequence is a solution.

<u>LEMMA 1.</u> Let C: $W \to X$ and B: $X \to Y$ be point-to-set maps. Suppose C is closed at $\vec{w}^\infty$, and B is closed on $C(\vec{w}^\infty)$. Also assume if $\vec{w}^k \to \vec{w}^\infty$, $k \in \kappa$, and if $\vec{x}^k \in C(\vec{w}^k)$, $k \in \kappa$, that for some $\kappa^1 \subset \kappa$

$$\vec{x}^k \to \vec{x}^\infty \qquad k \in \kappa^1 . \tag{4.33}$$

Then the composition $A = BC$ is closed at $\vec{w}^\infty$.

<u>COROLLARY 1.</u> Let C: $W \to X$ and B: $X \to Y$ be point-to-set maps. Suppose C is closed at $\vec{w}^\infty$ and B is closed on $C(\vec{w}^\infty)$. If X is compact, then $A = BC$: $W \to Y$ is closed at $\vec{w}^\infty$.

<u>COROLLARY 2.</u> Let C: $W \to X$ be a function and B: $X \to Y$ be a point-to-set map. Assume C is continuous at $\vec{w}^\infty$ and B is closed at $C(\vec{x}^\infty)$. Then the point-to-set map $A = BC$: $W \to Y$ is closed at $\vec{w}^\infty$.

The map M^1 represents a one-dimensional search. It minimizes the objective function on a segment either of the ray emanating from $\vec{x}^k$ in the direction $\vec{d}^k$ or of the line through $\vec{x}^k$ in the direction $\vec{d}^k$. Let $\vec{x}^{k+1}$ be the point produced by M^1. Mathematically

$$\vec{x}^{k+1} = \vec{x}^k + \tau^k \vec{d}^k \tag{4.34}$$

where

$$f(\vec{x}^{k+1}) = \min \{ f(\vec{x}^k + \tau \vec{d}^k) \mid \alpha \geq \tau \geq \beta \} \tag{4.35}$$

and α is either $+\infty$ or a positive scalar and, $\beta = 0, -\alpha$, or $-\infty$.

The map M^1: $E^{2n} \rightarrow E^n$ has the form

$$M^1(\vec{x},\vec{d}) = \{\vec{y} \mid f(\vec{y}) = \min_{\tau \varepsilon J} f(\vec{x} + \tau\vec{d}), \vec{y} = \vec{x} + \tau^0\vec{d}\} \qquad (4.36)$$

where $(\vec{x},\vec{d})$ is a point in E^{2n}, and J is an interval over which the scalar τ varies.

LEMMA 2. Let f be a continuous function. Then M^1 is closed if J is a closed and bounded interval.

DEFINITION 4. A mixed algorithm is an algorithm that has a given basic algorithm map B, which depends only upon $\vec{z}$, such that

$$B = A_k \qquad k \varepsilon \kappa . \qquad (4.37)$$

In other words, the basic map B is used infinitely often. For the remaining k, other maps are employed.

CONVERGENCE THEOREM B. Suppose there is an algorithmic map B: $V \rightarrow V$ for the nonlinear programming problems (with associated Z function and solution set Ω) that satisfies condition 1, 2, and 3 of Convergence Theorem A. Let a mixed algorithm for the problem be defined by the maps A_k: $V \rightarrow V$ such that for some κ

$$A_k = B \qquad k \varepsilon \kappa \qquad (4.38)$$

while for $k \notin \kappa$

$$Z(\vec{z}^{k+1}) \leq Z(\vec{z}^{k}) . \tag{4.39}$$

Further assume that

(1) All $\vec{z}^k \in X$ where X is compact, and

(2) If $\vec{z}^* \in \Omega$, and

$$Z(\vec{y}) \leq Z(\vec{z}^*) \tag{4.40}$$

then $$\vec{y} \in \Omega . \tag{4.41}$$

Then under these hypotheses the mixed algorithm either stops at a solution or generates a sequence $\{\vec{z}^k\}_1^\infty$ such that the limit of any convergent subsequence is a solution point.

<u>DEFINITION 5.</u> A step, given a nonoptimal point $\vec{x}$, that generates a point $\vec{y}$ for which

$$Z(\vec{y}) \leq Z(\vec{x}) \tag{4.42}$$

is called a spacer step. Also should $\vec{x}$ be a solution, then the spacer step must indicate this fact.

<u>DEFINITION 6.</u> A point-to-set map $A : V \to V$ is closed at $\vec{z}^\infty$

if $$\vec{z}^k \to \vec{z}^\infty, \quad \vec{z}^k \in A(\vec{z}^k), \quad \text{and} \quad \vec{y}^k \to \vec{y}^{\,\infty} \tag{4.43}$$

for $$k \in \kappa \quad \text{implies} \quad \vec{y}^\infty \in A(\vec{z}^\infty) . \tag{4.44}$$

The map is said to be closed on $X \subset V$ if it is closed at each $\vec{z} \in X$.

2. Mathematical Development

In light of the definitions, lemmas, corollaries, and theorems stated in the Mathematical Preliminaries, the GOOP algorithm is restated using the new terminology.

<u>GOOP ALGORITHM</u>

(1) Initialization step: $\vec{x}_1^0$ is given.

(2) Iteration k: Set i = 1.

(a) Calculate $\vec{d}_i$ using the Gram-Schmidt orthonormalization process.

(b) Calculate $\vec{x}_{i+1}^k \in M(\vec{x}_i^k, S)$ where M is the map defined below and $S = \{\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_i\}$.

(c) If $i = n$ go to step (3), otherwise set $i = i + 1$ and return to step (a).

(3) If $k = 0 \pmod{n}$ go to step (4), otherwise set $k = k + 1$ and return to step (2).

(4) Spacer step on $\vec{x}_{n+1}^k$ yielding $\vec{x}_{n+2}^k$.

(5) Set $\vec{x}_1^{k+1} = \vec{x}_{n+2}^k$, $k = k + 1$, and return to step (2).

The map M used in the GOOP algorithm is a mapping $M: E^{(i+1)n} \to E^n$. $M(\vec{x}_i^k, S)$ takes a point $\vec{x}_i^k$ and the subspace generated by the conjugate directions $\{\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_i\}$ and minimizes the function, J, in the generated subspace, i.e.,

$$M(\vec{x},S) = \{\vec{z} \,|\, J(\vec{z}) < J(\vec{x}), \text{ where } \vec{z} = \vec{x} + \vec{G}_i{}^T\vec{F}_0\vec{C}_i\}. \quad (4.45)$$

We can classify M as a one-dimensional search since the GOOP algorithm will continue to half the step, $\vec{G}_i{}^T\vec{F}_0\vec{C}_i$, until $J(\vec{x})$ has decreased.

In the description of the algorithm above, the spacer step has not been specified. The spacer step used a one-dimensional search in the direction $(\vec{x}^k_{n+1} - \vec{x}^k_1)$ which is the direction of the pattern move.

The proof of the convergence of the GOOP algorithm is based on the Convergence Theorem B. Before this theorem can be used, the convergence of the spacer step must be established. The following lemma establishes this convergence.

<u>LEMMA 3.</u> If the set X is compact and the objective function is continuous and has a unique minimum, then the algorithmic map $A = M^1D$, where M^1 represents a one-dimensional search, $D(\vec{x}) = (\vec{x},\vec{e})$, $\vec{e} = \vec{x}^k_{n+1} - \vec{x}^k_1$, is convergent. The interval for M^1 is $T = [-\alpha,\alpha]$.

<u>PROOF:</u> The map M^1D is closed by Corollary 2, because D is a continuous function and M^1 is a closed map (Lemma 2). By assumption all points are in a compact set. Then

$$A = M^1D: X \rightarrow X . \quad (4.46)$$

Then Corollary 1 verfies that the map A is closed as it is the composition of closed maps on compact sets, and condition 3 of Convergence Theorem A is verified.

To prove condition 2, we know that J has continuous first partial derivatives. By assumption, for any $\vec{x}$ and $\vec{e}$ there is a unique τ' for which

$$J(\vec{x} + \tau'\vec{e}) = \min_{\tau\varepsilon T} J(\vec{x} + \tau\vec{e}) . \tag{4.47}$$

Now let $\vec{z} = \vec{x}$ and $Z(\vec{z}) = J(\vec{x})$, and term a point $\vec{x}$ a solution if

$$\min_{\tau\varepsilon T} J(\vec{x} + \tau\vec{e}) = J(\vec{x}) \quad i = i,2,\ldots,n . \tag{4.48}$$

At such a point, because of the uniqueness assumption, $\vec{\nabla}J(\vec{x}) = 0$.

Condition 2 (a) holds easily because if $\vec{x}^k$ is not a solution

$$J(\vec{x}^{k+1}) < J(\vec{x}^k) . \tag{4.49}$$

Then by Theorem A, the algorithm $A = M^1D$ converges.//

Now taking the Mathematical Framework supplied by Zangwill and Lemma 3 just developed, the convergence of the GOOP-LAT algorithm can be examined. The following theorem establishes the convergence of the GOOP-LAT algorithm for the nonquadratic problem.

<u>THEOREM C.</u> Under the same assumptions stated in LEMMA 3, the GOOP-LAT algorithm, a mixed algorithm using the algorithmic map $A = M^1D$ as the spacer step, is convergent.

<u>PROOF:</u> By the previous Lemma 3, we know that the algorithmic map $A = M^1D$ (the spacer step) satisfies conditions 1, 2, and 3 of Convergence Theorem A. Then the map A would satisfy the condition in Theorem B for the map B. The mixed algorithm will be defined by the map $A_k = M^1D\colon V \to V$ such that for $\kappa = \{k | k = 0 \bmod (n+1)\}$

$$A_k = B \qquad k \in \kappa \tag{4.50}$$

while for $k \notin \kappa$

$$J(\vec{z}^{k+1}) \leq J(\vec{z}^{k}) . \tag{4.51}$$

The last fact follows from the property of conjugate direction algorithms. In generating the conjugate directions, $\vec{d}_1, \vec{d}_2, \ . \ . \ . \ , \vec{d}_i$, it follows from the definition of $M(\vec{x}_i^k, S)$, that

$$J(\vec{x}_{i+1}^k) \leq J(\vec{x}_i^k) . \tag{4.52}$$

Therefore, by the Convergence Theorem B, the GOOP-LAT algorithm is convergent.//

V. COMPUTATIONAL RESULTS

The linear acceleration technique (LAT) algorithm is used to solve eight traditional nonquadratic functions. The LAT algorithm is applied to three conjugate direction algorithms. The computational results of this chapter:

(1) demonstrate the improved convergence of the GOOP algorithm,

(2) demonstrate the effect of LAT on the convergence rate of other successful conjugate direction algorithms,

(3) demonstrate the efficiency of the GOOP algorithm performed in single precision arithmetic,

(4) demonstrate that the GOOP algorithm is a competitive algorithm with the most successful nonlinear programming algorithms.

In addition to previously referenced papers, many recent papers [34-39] have made comparisons of various algorithms and provide sample problems. From these studies, eight problems were chosen for comparison of algorithms. These eight problems are described in detail in appendix A.

The first problem is Rosenbrock's parabolic valley problem [2] with starting point (-1.2, 1.0). The second problem is CUBE attributed to Witte and Holst [40] with starting point (0.5, 0.5). Problem three is Beale's [41] problem with starting point (2.0, 0.7).

Problems four, five, and six are the four parameter problems attributed to Powell [42] with starting values (10.0,10.0,10.0,-10.0), (3.0,-1.0,0.0,1.0) and (-0.1,-0.1,0.1,0.1), respectively. The seventh problem is Wood's problem [43] with starting point (-3.0,-1.0,-3.0,-1.0). Problem eight is a four parameter problem attributed to Miele [44] with starting point (1.0,2.0,2.0,2.0).

A. Comparison Of Three Conjugate Direction Algorithms Using LAT With The Levenberg-Marquardt Algorithm

The following tables show results of the Levenberg-Marquardt algorithm and the conjugate direction algorithms: (1) GOOP, (2) Davidon-Fletcher-Powell, and (3) Fletcher-Reeves. The Levenberg-Marquardt algorithm is included for comparison because it has been found to be quite successful.

1. Description of Recorded Material

The following problems were performed on an IBM 360 Model 50 computer using double precision arithmetic except where stated otherwise.

Each table represents the results for a given problem for:

(1) Reset Fletcher-Reeves, FR(N).

(2) Reset Fletcher-Reeves, FR(N+1).

(3) Reset Fletcher-Reeves with LAT, FR(N)-LAT.

(4) Reset Fletcher-Reeves with LAT, FR(N+1)-LAT.

(5) Reset Davidon-Fletcher-Powell, DFP(N).

(6) Reset Davidon-Fletcher-Powell, DFP(N+1).

(7) Reset Davidon-Fletcher-Powell, with LAT, DFP(N)-LAT.

(8) Reset Davidon-Fletcher-Powell with LAT, DFP(N+1)-LAT.

(9) Levenberg-Marquardt Compromise,LMC.

(10) GOOP.

(11) GOOP with LAT, GOOP-LAT.

The two conjugate direction algorithms listed above are reset algorithms. The N or N+1 in parentheses indicates that the algorithm is reset after N iterations. The Fletcher-Reeves algorithm is restarted in the gradient direction. The Davidon-Fletcher-Powell algorithm resets the H-matrix to the identity matrix.

Because the Fletcher-Reeves and Davidon-Fletcher-Powell algorithms are different from the GOOP and the Levenberg-Marquardt algorithms in form, different stopping conditions are used for the two types of algorithms. In comparing the results generated for the eight problems using the two types of algorithms, it is believed that the stopping conditions listed below produce a reasonable comparison. Algorithms (1)-(8) use the stopping condition

$$\vec{g}_i^{\,T}\vec{g}_i \leq 10^{-12} \; . \tag{5.1}$$

For algorithms (9)-(11), the stopping condition is

$$f_i \leq 10^{-8} . \tag{5.2}$$

In each table, the information provided is: (1) number of iterations, IT, (2) function evaluation, FE, and (3) value of the function when the algorithm terminated, V of F. The number of iterations is a common comparative item in the literature. For the methods being compared in this paper, the number of iterations does not give a true comparison. Specifically, the GOOP algorithm and the Levenberg-Marquardt algorithm solve the Gauss equation (2.7). The Fletcher-Reeves and Davidon-Fletcher-Powell algorithms solve the Newton-Raphson equation (2.32). The GOOP, Fletcher-Reeves, and Davidon-Fletcher-Powell algorithms optimize in the n-conjugate directions. Therefore, it is felt that the number of function evaluations is more informative about the relative merits of the various algorithms. To provide additional information about the progress of the algorithm, the value of the function when the algorithm terminated is also indicated.

In the LAT algorithm, a one-dimensional search is used. Both the Fletcher-Reeves and Davidon-Fletcher-Powell algorithms use the cubic interpolation search in optimizing each iteration and this search could be used in LAT. However, since no one-dimensional search is used in GOOP and Levenberg-Marquardt algorithms, a simple one-dimensional search has been written and used in LAT for all algorithms above. A flow chart for this simple one-dimensional search is found in appendix B.

2. Numerical Results on Comparison of Algorithms

TABLE 1

PROBLEM 1

Algorithm	IT	FE	V of F
FR(N)	48	101	$0.1037 \quad 10^{-16}$
FR(N+1)	30	64	$0.2659 \quad 10^{-12}$
FR(N)-LAT	19	70	$0.1028 \quad 10^{-12}$
FR(N+1)-LAT	30	91	$0.1505 \quad 10^{-12}$
DFP(N)	32	79	$0.1462 \quad 10^{-20}$
DFP(N+1)	35	90	$0.1189 \quad 10^{-15}$
DFP(N)-LAT	14	60	$0.7402 \quad 10^{-17}$
DFP(N+1)-LAT	25	89	$0.4949 \quad 10^{-13}$
GOOP	64	224	$0.6985 \quad 10^{-10}$
GOOP-LAT	24	121	$0.4020 \quad 10^{-10}$
LMC	23	88	$0.1812 \quad 10^{-8}$

TABLE 2

PROBLEM 2

Algorithm	IT	FE	V of F	
FR(N)	25	56	0.8283	10^{-21}
FR(N+1)	18	44	0.3125	10^{-15}
FR(N)-LAT	19	71	0.3197	10^{-13}
FR(N+1)-LAT	18	62	0.3125	10^{-15}
DFP(N)	23	60	0.1602	10^{-19}
DFP(N+1)	20	64	0.2527	10^{-16}
DFP(N)-LAT	13	51	0.6687	10^{-15}
DFP(N+1)-LAT	18	76	0.1131	10^{-16}
GOOP	46	161	0.3881	10^{-9}
GOOP-LAT	20	106	0.1211	10^{-9}
LMC	6	20	0.3773	10^{-12}

TABLE 3

PROBLEM 3

Algorithm	IT	FE	V of F	
FR(N)	9	22	0.7820	10^{-13}
FR(N+1)	11	26	0.8987	10^{-16}
FR(N)-LAT	9	34	0.7820	10^{-13}
FR(N+1)-LAT	11	35	0.8987	10^{-16}
DFP(N)	9	24	0.2131	10^{-15}
DFP(N+1)	11	32	0.5083	10^{-14}
DFP(N)-LAT	9	36	0.2131	10^{-15}
DFP(N+1)-LAT	11	41	0.5083	10^{-14}
GOOP	18	63	0.8174	10^{-9}
GOOP-LAT	10	50	0.6725	10^{-9}
LMC	5	15	0.2792	10^{-17}

TABLE 4

PROBLEM 4

Algorithm	IT	FE	V of F	
FR(N)	74	167	0.9565	10^{-9}
FR(N+1)	50	123	0.4782	10^{-9}
FR(N)-LAT	58	182	0.3682	10^{-9}
FR(N+1)-LAT	40	129	0.5406	10^{-10}
DFP(N)	50	204	0.6842	10^{-9}
DFP(N+1)	29	100	0.1341	10^{-9}
DFP(N)-LAT	39	182	0.1251	10^{-9}
DFP(N+1)-LAT	34	154	0.1448	10^{-9}
GOOP	24	156	0.2049	10^{-8}
GOOP-LAT	18	136	0.1202	10^{-8}
LMC	15	57	0.6832	10^{-11}

TABLE 5

PROBLEM 5

Algorithm	IT	FE	V of F	
FR(N)	106	216	0.3325	10^{-10}
FR(N+1)	57	121	0.5196	10^{-10}
FR(N)-LAT	42	128	0.2927	10^{-10}
FR(N+1)-LAT	47	130	0.7912	10^{-10}
DFP(N)	26	96	0.8097	10^{-10}
DFP(N+1)	36	166	0.2400	10^{-9}
DFP(N)-LAT	25	110	0.7763	10^{-10}
DFP(N+1)-LAT	42	230	0.1035	10^{-9}
GOOP	14	91	0.8549	10^{-9}
GOOP-LAT	12	100	0.1326	10^{-8}
LMC	9	27	0.2367	10^{-8}

TABLE 6

PROBLEM 6

Algorithm	IT	FE	V of F	
FR(N)	25	57	0.1262	10^{-10}
FR(N+1)	26	58	0.3840	10^{-9}
FR(N)-LAT	25	74	0.1888	10^{-11}
FR(N+1)-LAT	37	108	0.2147	10^{-9}
DFP(N)	34	153	0.1045	10^{-8}
DFP(N+1)	14	54	0.9195	10^{-12}
DFP(N)-LAT	34	177	0.1045	10^{-8}
DFP(N+1)-LAT	14	60	0.9195	10^{-12}
GOOP	14	91	0.3156	10^{-8}
GOOP-LAT	8	61	0.7246	10^{-9}
LMC	6	18	0.1448	10^{-8}

TABLE 7

PROBLEM 7

Algorithm	IT	FE	V of F	
FR(N)	38	81	0.1717	10^{-14}
FR(N+1)	28	61	0.1635	10^{-14}
FR(N)-LAT	35	102	0.4289	10^{-14}
FR(N+1)-LAT	28	76	0.1635	10^{-14}
DFP(N)	57	129	0.6472	10^{-16}
DFP(N+1)	56	131	0.3411	10^{-13}
DFP(N)-LAT	52	160	0.3377	10^{-22}
DFP(N+1)-LAT	50	145	0.4512	10^{-18}
GOOP	> 80	---	------	--
GOOP-LAT	58	445	0.8781	10^{-11}
LMC	53	208	0.9870	10^{-16}

TABLE 8

PROBLEM 8

Algorithm	IT	FE	V of F	
FR(N)	57	117	0.3350	10^{-9}
FR(N+1)	82	167	0.3452	10^{-8}
FR(N)-LAT	45	127	0.2686	10^{-8}
FR(N+1)-LAT	46	130	0.2810	10^{-8}
DFP(N)	53	363	0.4864	10^{-8}
DFP(N+1)	33	180	0.6752	10^{-8}
DFP(N)-LAT	30	186	0.2760	10^{-8}
DFP(N+1)-LAT	33	198	0.6752	10^{-8}
GOOP	20	134	0.9014	10^{-8}
GOOP-LAT	10	80	0.2665	10^{-8}
LMC	20	75	0.6988	10^{-10}

B. Comparison of GOOP Algorithms Using Single and Double Precision Arithmetic

While compiling the results of the previous section, all problems for each algorithm were computed using single precision and double precision. In the case of the Fletcher-Reeves and Davidon-Fletcher-Reeves algorithms, the problems solved in double precision definitely improved the convergence rate. The number of iterations and number of function evaluations decreased in almost every problem. However, in the case of the GOOP and the Levenberg-Marquardt algorithms, the convergence rate was not improved when using double precision arithmetic.

1. Description of Recorded Material

Each table represents the results for a given problem of:

(1) GOOP in single precision, GOOP (S)

(2) GOOP-LAT in single precision, GOOP-LAT (S)

(3) GOOP in double precision, GOOP (D)

(4) GOOP-LAT in double precision, GOOP-LAT (D)

The stopping conditions and one-dimensional search used in the previous section are the same in the following tables.

2. Numerical Results On The Effect Of Precision

TABLE 9

PROBLEM 1

Algorithm	IT	FE	V of F
GOOP (S)	64	160	$0.6040 \quad 10^{-10}$
GOOP-LAT (S)	28	115	$0.2628 \quad 10^{-8}$
GOOP (D)	64	224	$0.6985 \quad 10^{-10}$
GOOP-LAT (D)	24	121	$0.4020 \quad 10^{-10}$

TABLE 10

PROBLEM 2

Algorithm	IT	FE	V of F
GOOP (S)	46	161	$0.2785 \quad 10^{-9}$
GOOP-LAT (S)	12	65	$0.7994 \quad 10^{-10}$
GOOP (D)	46	161	$0.3881 \quad 10^{-9}$
GOOP-LAT (D)	20	106	$0.1211 \quad 10^{-9}$

TABLE 11

PROBLEM 3

Algorithm	IT	FE	V of F
GOOP (S)	18	63	0.1103 10^{-8}
GOOP-LAT (S)	10	50	0.1103 10^{-8}
GOOP (D)	18	63	0.8174 10^{-9}
GOOP-LAT (D)	10	50	0.6725 10^{-9}

TABLE 12

PROBLEM 4

Algorithm	IT	FE	V of F
GOOP (S)	24	156	0.1254 10^{-8}
GOOP-LAT (S)	16	120	0.2936 10^{-9}
GOOP (D)	24	156	0.2049 10^{-8}
GOOP-LAT (D)	18	136	0.1202 10^{-8}

TABLE 13

PROBLEM 5

Algorithm	IT	FE	V of F
GOOP (S)	14	91	0.3705 10^{-8}
GOOP-LAT (S)	12	100	0.1458 10^{-8}
GOOP (D)	14	91	0.8549 10^{-9}
GOOP-LAT (D)	12	100	0.1326 10^{-8}

TABLE 14

PROBLEM 6

Algorithm	IT	FE	V of F	
GOOP (S)	14	91	0.2968	10^{-8}
GOOP-LAT (S)	8	62	0.2632	10^{-10}
GOOP (D)	14	91	0.3156	10^{-8}
GOOP-LAT (D)	8	61	0.1448	10^{-8}

TABLE 15

PROBLEM 7

Algorithm	IT	FE	V of F	
GOOP (S)	> 80	---	------	--
GOOP-LAT (S)	50	389	0.1076	10^{-10}
GOOP (D)	> 80	---	------	--
GOOP-LAT (D)	58	445	0.8781	10^{-11}

TABLE 16

PROBLEM 8

Algorithm	IT	FE	V of F	
GOOP (S)	20	134	0.8891	10^{-8}
GOOP-LAT (S)	10	83	0.3369	10^{-10}
GOOP (D)	20	134	0.9014	10^{-8}
GOOP-LAT (D)	10	80	0.2665	10^{-8}

C. Operations Count for Function and Derivative Evaluation

In examining tables 1-8, it appears that the GOOP algorithm compares poorly with the other algorithms. It must be noted that the GOOP algorithm does not use first derivatives, but rather the algorithm uses approximations of the derivatives and only calculates the derivatives for one column of the F matrix per iteration. Therefore, a comparison of function evaluation is not valid. In this section, a more valid comparative procedure is attempted. For each problem used in section A, an operations count is made for each function and derivative evaluation.

1. Procedure for Making Operations Count

In comparing algorithms, one means of comparison is the amount of arithmetic to perform the algorithms. The sources of arithmetic are: (1) execution of the algorithm, (2) evaluation of the function, and (3) evaluation of the derivatives. In this paper, the number of operations to evaluate the function and derivatives are compared for the specific problems solved.

Because the four algorithms (FR, DFP, LMC, and GOOP) use the derivatives differently, it is difficult to make a valid comparison of total number of operations for evaluting the function and derivatives. The procedure for making the count is described below, but the details of the actual count for each problem is supplied in appendix C.

The Fletcher-Reeves and Davidon-Fletcher-Powell algorithms evaluate the gradient at the same time that the function is evaluated. Thus the total number of operations performed to evaluate the function and gradient once will be multiplied by the function evaluations, FE, for a given problem. This will give the total number of operations for function and derivative evaluation for this problem.

The Levenberg-Marquardt algorithm makes function evaluations and uses the derivatives in constructing the H matrix of equation (1.8). For each iteration the H^TH matrix and $H^T\vec{h}_0$ vector are constructed and then scaled. If H has dimension m x n and one observes that H^TH is symmetric, then H^TH is constructed using $(n+1)nm/2$ multiplications and $(n+1)n(m-1)/2$ additions. To construct $H^T\vec{h}_0$, m multiplications and m-1 additions are used. Since H^TH is symmetric, scaling H^TH requires one multiplication and one division for each element of H^TH above the main diagonal. This would use $(n-1)n$ multiplications and divisions. To scale $H^T\vec{h}_0$, one division per element is needed, i.e., n divisions. The total operations count for the Levenberg-Marquardt algorithm is the number of operations to construct and scale H^TH and $H^T\vec{h}_0$, multiplied by the number of iterations, IT, plus the number of operations to evaluate the function multiplied by the number of function evaluations, FE.

2. Description of Recorded Material

The procedure described above is applied to each of the eight problems used in section A. The number of multiplications and divisions, MD; additions and subtractions, AS; exponential subprogram, EXP; tangent subprogram, TAN; and cosine subprogram, COS; performed are recorded for each algorithm and problem. The details of how each problem was evaluated is found in appendix C.

3. Numerical Results

TABLE 17

OPERATIONS COUNT

Problem	Algorithm	MD	AS	EXP	TAN	COS
1	FR	512	256			
	DFP	480	240			
	LMC	628	333			
	GOOP	516	401			
2	FR	396	176			
	DFP	459	204			
	LMC	172	78			
	GOOP	349	219			
3	FR	440	264			
	DFP	480	288			
	LMC	215	160			
	GOOP	430	430			
4	FR	2214	1353			
	DFP	1800	1100			
	LMC	1500	894			
	GOOP	1264	904			
5	FR	2178	1331			
	DFP	1728	1056			
	LMC	828	486			
	GOOP	966	693			
6	FR	1026	627			
	DFP	972	594			
	LMC	552	324			
	GOOP	642	459			
7	FR	1342	1159			
	DFP	2838	2451			
	LMC	7739	6202			
	GOOP	5407	5407			
8	FR	3042	936	117	117	117
	DFP	4680	1440	180	180	180
	LMC	2460	1180	75	75	20
	GOOP	1010	370	80	80	0

D. Summary of Computational Results

From the computational results of this chapter, the following comments are in order.

1. Improved Convergence Rate of the GOOP Algorithm

In all of the eight problems presented, the number of iterations for the GOOP-LAT algorithm is less than the number of iterations for the GOOP algorithm. In one of the eight problems, problem 5, the number of function evaluations is less for the GOOP algorithm. This would imply that for this particular problem, the addition of the LAT algorithm is inefficient (causing more function evaluations). Closer examination of the results of this problem, show that when the one-dimensional search is performed, the optimal point obtained is near the $\vec{b}_2$ vector defined in chapter 3, section A. The simple one-dimensional search used in LAT requires a large number of function evaluations to obtain this type of optimal point. However the algorithm is making a relatively small improvement toward the minimal point of the function. A more efficient one-dimensional search in LAT should produce better results for this particular problem.

2. Effect of LAT on the Convergence Rate of the FR Algorithm

In almost every problem, the number of iterations for

FR-LAT is less than or equal to the number of iterations for FR. But in approximately fifty per cent of the cases presented in tables 1-8, the LAT algorithm does not improve the rate of convergence. Also when the number of iterations is decreased, the number of function evaluations does not decrease. In this case, the FR-LAT algorithm will not be as efficient as the FR algorithm.

3. Effect of LAT on the Convergence Rate of the DFP Algorithm

Again in almost every problem, the number of iterations for DFP-LAT is less than or equal to the number of iterations for DFP. As in the case of the FR algorithm, in approximately fifty per cent of the cases presented in tables 1-8, the LAT algorithm does not improve the rate of convergence. However, for several problems an improvement of the rate of convergence (decrease in iterations) and a decrease in function evaluations is found. Problems 1 and 2 demonstrate these properties.

4. Comparison of Single Precision and Double Precision Arithmetic for the GOOP Algorithm

Tables 9-16 indicate that the GOOP and GOOP-LAT algorithms make very little if any improvement in convergence rate by using double precision arithmetic. In fact, for some problems described, single precision arithmetic results are better than the double precision arithmetic results. Although the results for single precision arithmetic of the FR and DFP algorithms are not provided in this thesis, it was found that these algorithms had to be performed in double precision arithmetic to obtain accurate results. Yet for the GOOP and LMC algorithms, the double precision arithmetic was not as necessary.

5. Comparison of the Four Algorithms on the Basis of an Operations Count

Since the GOOP algorithms must approximate the derivatives, the new comparative procedure of comparing an operations count for function and derivative evaluations is found in table 17. It appears that in problem 2, 3, and 7 the GOOP algorithm compares unfavorably with the optimal algorithm. In problems 1, 4, 5, and 6 the GOOP algorithm is comparable with the optimal algorithm. In problem 8, the GOOP algorithm is much more efficient. Also note that no single algorithm performed well for all problems.

One other comparison of the GOOP algorithm should be with a successful version of the DFP algorithm which approximates derivatives by Stewart [9]. In appendix D the results for problems 1 and 5 have been reproduced from Stewart's paper. In this comparison, the GOOP algorithm appears quite favorably. In problem 1, Stewart's algorithm performed 145 function evaluations for $f = 2.8 \times 10^{-8}$ while GOOP performed 115 function evaluations for $f = 2.6 \times 10^{-9}$. In problem 5, Stewart's algorithm performed 139 function evaluations for $f = 1.1 \times 10^{-8}$ while GOOP performed 91 function evaluations for $f = 3.7 \times 10^{-9}$.

VI. CONCLUSIONS

In some applied fields, the GOOP algorithm has been successfully used, but very little theoretical work has been developed for this algorithm. This thesis adds the linear acceleration technique, LAT, to the GOOP algorithm. It then provides the theoretical demonstration which had been conjectured to be true by many users, that the GOOP-LAT algorithm is convergent for the nonquadratic nonlinear programming problem. This thesis does not consider the problem of examining the rate of convergence, but it is conjectured that the GOOP-LAT algorithm has a rate of convergence somewhere between linear and quadratic.

The linear acceleration technique, LAT, described in chapter III definitely makes an improvement on the convergence rate of the GOOP algorithm. In all the problems of chapter V, the number of iterations decrease. In all but one problem, the number of function evaluations also decreases. Thus the GOOP-LAT algorithm does improve the convergence rate of the GOOP algorithm and because the number of function evaluations decreased, the new algorithm is more efficient than the old.

In applying LAT to the conjugate direction algorithms, FR and DFP, the improvement of the convergence rate is not as outstanding. In almost all of the problems, the convergence rate is improved or unchanged. However, in most cases the addition of the LAT algorithm is not efficient. Even though the number of iterations decreased, the number of function evaluations increased.

In comparing the GOOP algorithms with the Levenberg-Marquardt algorithm and other conjugate direction algorithms, the GOOP-LAT algorithm compares more favorably than the GOOP algorithm for most problems. The Levenberg-Marquardt algorithm produced good results if the number of elements of the vector $\vec{h}$, equation (1.5), is small and if the elements of $\vec{h}$ have partial derivatives which have a small number of operations. The Fletcher-Reeves and Davidon-Fletcher-Powell algorithms produced good results if the gradient vector, equation (1.2), has a small number of operations. The GOOP-LAT algorithm produced favorable results on several problems, but the best results were for a problem where the number of elements of the vector $\vec{h}$ was large and the partial derivatives of these elements were not relatively simple.

One other attribute of the GOOP-LAT algorithm is that the single precision arithmetic computational results are just as favorable as the double precision results. This is not an attribute of the Fletcher-Reeves and Davidon-Fletcher-Powell algorithms.

It is felt that the numerical results of this thesis indicate that the GOOP-LAT algorithm in single precision arithmetic is an algorithm which is as efficient an algorithm as the other successful algorithms presented. It must be noted that no algorithm is going to solve all problems efficiently, but the GOOP-LAT algorithm is competitive with these algorithms. The GOOP-LAT algorithm should be used for the following types of problems: (1) those problems with partial derivatives which are extremely difficult to obtain; (2) those problems where the number of elements of the vector $\vec{h}$ are large and a large number of operations are necessary to evaluate the partial derivatives. It is conjectured that a more efficient one-dimensional search in the LAT algorithm might improve the convergence rate of the GOOP-LAT algorithm even more.

The work described in this paper has been concerned with the development of the LAT algorithm, its application to conjugate direction algorithms, the convergence of the GOOP-LAT algorithm, and the comparison of the GOOP-LAT algorithm with other successful algorithms. Two specific extensions of this work appear to warrant further study. The LAT algorithm should be constructed using a more sophisticated one-dimensional search and the rate of convergence for the GOOP-LAT algorithm should be examined.

McCormick [29] has established that the reset Davidon-Fletcher-Powell algorithm is superlinear convergent.* It is conjectured that the GOOP-LAT algorithm has superlinear convergence.

---

*Definition of Superlinear Convergence. An algorithm's convergence is superlinear with respect to every n points within a sequence if

$$\lim_{c\to\infty} \frac{||\vec{x}_c^n - \vec{x}^*||}{||\vec{x}_c^0 - \vec{x}^*||} = 0$$

where $\vec{x}^*$ is the optimal point of the objective function and $\vec{x}_j^i$ indicates the ith point of the sequence which has been reset j times.

BIBLIOGRAPHY

1. Hooke, R. and Jeeves, T. A. "Direct Search Solution of Numerical and Statistical Problems," Journal of the Association for Computing Machinery, Vol. 8, 1961, pp. 212-229.

2. Rosenbrock, H. H. "An Automatic Method for Finding the Greatest or Least Value of a Function," The Computer Journal, Vol. 3, No. 2, 1960, p. 175.

3. Grey, D. S. "Aberration Theories for Semi-Automatic Lens Design by Electronic Computers. I. Preliminary Remarks," Journal of Optical Society of America, Vol. 53, No. 6, 1963, pp. 672-676.

4. Grey, D. S. "Aberration Theories for Semi-Automatic Lens Design by Electronic Computers. II. A Specific Computer Program," Journal of Optical Society of America, Vol. 53, No. 6, 1963, pp. 677-680.

5. Pegis, R. J.; Grey, D. S.; Vogl, T. P.; and Rigler, A. K. "The Generalized Orthonormal Optimization Program and Its Application," Recent Advances in Optimization Techniques, ed. by A. Lavi and T. P. Vogal. New York: John Wiley and Sons, 1966.

6. Fletcher, R. and Reeves, C. M. "Function Minimization by Conjugate Gradients," The Computer Journal, Vol. 7, 1959, pp. 149-153.

7. Davidon, W. C. "Variable Metric Method for Minimization," A.E.C. Research and Development Report, ANL-5990, 1959.

8. Fletcher, R. and Powell, M. J. D. "A Rapidly Convergent Descent Method for Minimization," The Computer Journal, Vol. 7, 1964, pp. 163-168.

9. Stewart, G. W. "A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives," Journal of the Association of Computing Machinery, Vol. 4, 1962, pp. 441-452.

10. Huang, H. Y. "Unified Approach to Quadratically Convergent Algorithms for Function Minimization," Journal of Optimization Theory and Applications, Vol. 4, No. 5, 1970, pp. 405-423.

11. Gauss, K. F. Theoria Motus Corporum Coelestiam, 1809.

12. Hartley, H. O. "The Modified Gauss-Newton Method for the Fitting of Non-Linear Regression Functions by Least Squares," Technometrics, Vol. 3, No. 2, 1961, pp. 269-280.

13. Levenberg, K. "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Quarterly of Applied Mathematics, Vol. 8, 1944, pp. 212-221.

14. Marquardt, Donald W. "An Algorithm for Least-Squares Estimation of Non-Linear Parameters," Journal of the Society of Industrial Applied Mathematics, Vol. 2, 1963, pp. 164-168.

15. Jones, A. "Spiral--A New Algorithm for Non-Linear Parameter Estimation Using Least Squares," The Computer Journal, Vol. 13, 1970, pp. 301-308.

16. Meyer, R. R. "Theoretical and Computational Aspects of Nonlinear Regression," Nonlinear Programming, ed. by J. B. Rosen, O. L. Mangasarin, and K. Ritter. New York: Academic Press, 1970, pp. 465-485.

17. Broste, N. A. and Lavi, A. An Orthonormal Optimization Technique: Theory, Implementation and Application. Pittsburgh: Carnegie-Mellon University, 1968.

18. Powell, M. J. D. "A Survey of Numerical Methods for Unconstrained Optimization," SIAM Review, Vol. 12, No. 1, 1970, pp. 79-97.

19. Fletcher, R. "A Review of Methods for Unconstrained Optimization," Optimization, ed. by R. Fletcher. New York: Academic Press, 1969, pp. 1-12.

20. Broyden, C. G. "A Class of Methods for Solving Nonlinear Simultaneous Equations," Mathematics of Computations, Vol. 19, No. 92, 1965, pp. 577-593.

21. Zeleznik, Frank J. "Quasi-Newton Methods for Nonlinear Equations," Journal of the Association for Computing Machinery, Vol. 15, No. 2, 1968, pp. 265-271.

22. Adachi, Norihiko. "On Variable-Metric Algorithms," Journal of Optimization Theory and Applications, Vol. 7, No. 6, 1971, pp. 391-410.

23. Cauchy, A. "Methods Générale Pour La Résolution Des Systems D`équations Simultanées," C. R. Acad. Sci. Paris, No. 25, 1847, pp. 368-381.

24. Forsythe, G. E. "On the Asymptotic Directions of the s-Dimensional Optimum Gradient Method, Numerische Mathematik, Vol. 11, 1968, pp. 57-76.

25. Hestenes, M. R. and Stiefel, E. "Methods of Conjugate Gradients for Solving Linear Systems," Journal of Research, National Bureau of Standards, Vol. 49, 1961, pp. 212-221.

26. Shah, B. V.; Buehler, R. J.; and Kempthorne, O. "Some Algorithms for Minimizing a Function of Several Variables," Journal of the Society of Industrial Mathematics, Vol. 12, 1964, pp. 74-92.

27. Powell, M. J. D. "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives," The Computer Journal, Vol. 3, 1964, pp. 155-162.

28. Huang, H. Y. "Numerical Experiments on Quadratically Convergent Algorithms for Function Minimization," Journal of Optimization Theory and Applications, Vol. 6, 1970, pp. 269-282.

29. McCormick, G. P. The Rate of Convergence of the Reset Davidon Variable Metric Method. Madison, Wisconsin: The University of Wisconsin, Mathematics Research Center, United States Army, MRC Technical Summary Report No. 1012, 1969.

30. Broyden, C. G. "Quasi-Newton Methods and Their Application to Function Minimization," Mathematics Computations, Vol. 21, 1967, pp. 368-381.

31. Goldfarb, D. "Sufficient Conditions for the Convergence of a Variable Metric Algorithm," Optimization, ed. by R. Fletcher. New York: Academic Press, 1969, pp. 273-281.

32. Buchele, Donald R. "Damping Factor for the Least-Squares Method of Optical Design," Applied Optics, Vol. 7, No. 12, pp. 2433-2435.

33. Zangwill, Willard I. Nonlinear Programming: A Unified Approach. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1969.

34. Colville, A. R. A Comparative Study of Nonlinear Programming Codes. New York: IBM New York Scientific Center Report No. 320-2949, 1968.

35. Cox, R. A. Comparison of the Performance of Seven Optimization Algorithms on Twelve Unconstrained Optimization Problems. Pittsburgh: Gulf Research and Development Company Ref. 1335CN04, 1969.

36. Holzman, A. G. Comparitive Analysis of Nonlinear Programming Codes with the Weisman Algorithm. Pittsburgh: University of Pittsburgh Space Research Coordinate Center Report No. 113, 1969.

37. Leon, A. "A Comparison Among Eight Known Optimizing Procedures," Recent Advances in Optimization Techniques, ed. by A. Lavi and T. P. Vogl. New York: John Wiley and Sons, 1966, pp. 23-46.

38. Murtagh, B. A. and Sargent, R. W. H. "Computational Experience with Quadratically Convergent Minimization Methods," The Computer Journal, Vol. 13, No. 2, 1970, pp. 185-194.

39. Shanno, D. F. "Parameter Selection for Modified Newton Methods for Function Minimization," SIAM Journal of Numerical Analysis, Vol. 7, No. 3, 1970, pp. 368-372.

40. Witte, B. F. and Holst, W. R. "Two New Direct Minimum Search Procedures for Functions of Several Variables," submitted for presentation at the 1964 Spring Joint Computer Conference in Washington, D. C.

41. Beale, E. M. L. On an Iterative Method for Finding a Local Minimum of a Function of More Than One Variable. Princeton: Technical Report 25, Statistical Techniques Research Group, Princeton University, 1958.

42. Powell, M. J. D. "An Iterative Method for Finding Stationary Values of a Function of Several Variables," The Computer Journal, Vol. 5, p. 147.

43. Pearson, J. D. On Variable Metric Methods of Minimization. McLean, Virginia: Research Analysis Corporation, Technical Paper No. RAC-TP-302, 1968.

44. Cragg, E. E. and Levy, A. V. "Study on a Supermemory Gradient Method for the Minimization of Functions," Journal of Optimization Theory and Application, Vol. 4, No. 3, 1969, pp. 191-205.

VITA

Larry W. Cornwell was born on July 29, 1941, in Quincy, Illinois. He received his secondary education at Unity High School in Mendon, Illinois. After completing four years at Culver-Stockton College in Canton, Missouri, he received his Bachelor of Arts degree with a major in mathematics.

From September 1963 to June 1965, he attended Southern Illinois University in Carbondale, Illinois, whereupon he received his Master of Science in Education degree with a major in mathematics. During the academic year 1963-1964, he taught in the Mathematics Department as a graduate assistant. He taught mathematics at Murphysboro High School in Murphysboro, Illinois during the academic year 1964-1965.

From 1965 to 1966, he was an Instructor of Mathematics at Forest Park Community College in St. Louis, Missouri. He then served as Assistant Professor and Head of the Mathematics Department at Culver-Stockton College from 1966 to 1969. During the summers of 1967, 1968, and 1969, he was granted an N.S.F. Fellowship to attend a Computer Science Institute at the University of Missouri-Rolla.

From June 1969 to September 1971, he attended the University of Missouri-Rolla. During the academic year 1969-1970, he taught in the Computer Science Department as a graduate assistant. The following year he was granted a N.S.F. Science Faculty Fellowship. He is presently employed as Assistant Professor of Mathematics at Western Illinois University in Macomb, Illinois.

On June 15, 1963 he was married to the former Sara K. Finke of Quincy, Illinois. They have three children, Brent, Krista, and Michelle.

APPENDIX A

Description of Problems 1-8

Problem 1 (Rosenbrock's Function)

$$f = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

where the minimum f = 0 is found at the point (1.0,1.0).

Problem 2 (Cube)

$$f = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$$

where the minimum f = 0 is found at the point (1.0,1.0).

Problem 3 (Beale's Function)

$$f = (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2$$

where the minimum f = 0 is found at the point (3.0,0.5).

Problem 4 (Powell's Four Parameter Function)

$$f = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^2 + 10(x_1 - x_4)^2$$

where the minimum f = 0 is found at the point (0.0,0.0,0.0,0.0).

Problem 5

Same function as problem 4.

Problem 6

Same function as problem 4.

Problem 7 (Wood's Function)

$$f = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 0.2(x_2 - 1)^2 + 0.2(x_4 - 1)^2 + 9.9(x_2 + x_4 - 2)^2$$

which is equivalent to the more familiar form

$$f = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1(x_2 - 1)^2 + 10.1(x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$$

where the minimum f = 0 is found at the point (1.0,1.0,1.0,1.0).

Problem 8 (Miele Function)

$$f = (e^{x_1} - x_2)^4 + 100(x_2 - x_3)^6 + \tan^4(x_3 - x_4) + x_1^8 + (x_4 - 1)^2$$

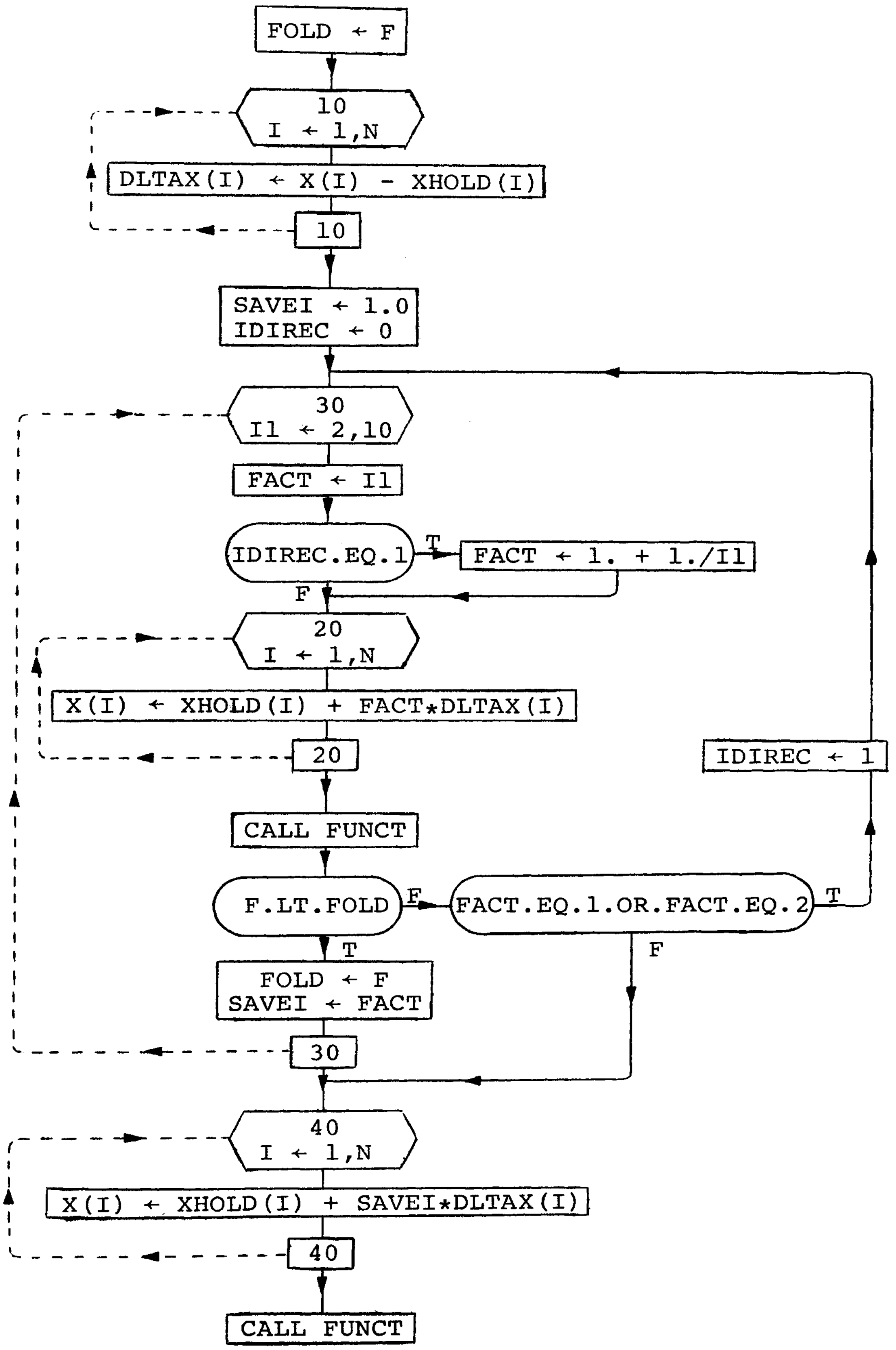
where the minimum f = 0 is found at the point (0.0,1.0,1.0,1.0).

APPENDIX B

FORTRAN Flow Chart of One-Dimensional Search

In the flow chart below, the following variables and subprogram are used

FUNCT	- Function subprogram which determines the value of the function, F, for the current value of the vector X.
F	- Current value of the function.
FOLD	- Minimum value of the function.
XHOLD	- Vector of $\vec{x}$-values at the beginning of the conjugate direction algorithm.
X	- Vector of current $\vec{x}$-values.
DLTAX	- Vector indicating direction determined by the conjugate direction algorithm.
IDIREC	- Flag. If IDIREC = 0, search is adding multiples of DLTAX. If IDIREC = 1, search is adding increments of DLTAX.
FACT	- Amount of DLTAX being added to base point XHOLD.
SAVEI	- Holds last successful value of FACT.


FOLD ← F
10
I ← 1,N
DLTAX(I) ← X(I) - XHOLD(I)
10
SAVEI ← 1.0
IDIREC ← 0
30
I1 ← 2,10
FACT ← I1
IDIREC.EQ.1
T
FACT ← 1. + 1./I1
F
20
I ← 1,N
X(I) ← XHOLD(I) + FACT*DLTAX(I)
20
IDIREC ← 1
CALL FUNCT
F.LT.FOLD
F
FACT.EQ.1.OR.FACT.EQ.2
T
T
F
FOLD ← F
SAVEI ← FACT
30
40
I ← 1,N
X(I) ← XHOLD(I) + SAVEI*DLTAX(I)
40
CALL FUNCT

APPENDIX C

Computational Details of the Operations Count

The procedure described in chapter V, section C is applied to the eight problems described in appendix A. The values for IT and FE used below are the most optimal values for each particular algorithm found in tables 1-16. The following abbreviations are used: multiplication and division, MD; addition and subtraction, AS; exponentiation subprogram, EXP; tangent subprogram, TAN; and cosine subprogram, COS. In evaluating gradients and derivatives, operations are only counted for terms which have not previously been computed in the function and derivative evaluation earlier.

Problem 1

a) Evaluation of function

MD = 4
AS = 3

b) The gradient is

$$\vec{g} = \begin{bmatrix} -400x_1(x_2 - x_1^2) + 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}$$

Evaluation of gradient

MD = 4
AS = 1

Total number of operations for function and gradient

MD = 8
AS = 4

c) The H matrix is

$$\begin{bmatrix} -20x_1 & 10 \\ -1 & 0 \end{bmatrix}$$

For construction of H

MD = 1
AS = 0

For $H^T H$ and $H^T \vec{h}_0$

MD = 8
AS = 3

For scaling $H^T H$ and $H^T \vec{h}_0$

MD = 3
AS = 0

Total number of operations for construction and scaling of $H^T H$ and $H^T \vec{h}_0$

MD = 12
AS = 3

<u>FR Algorithm</u>

MD = 8FE = 8(64) = 512
AS = 4FE = 4(64) = 256

<u>DFP Algorithm</u>

MD = 8FE = 8(60) = 480
AS = 4FE = 4(60) = 240

<u>LMC Algorithm</u>

MD = 12IT + 4FE = 12(23) + 4(88) = 628
AS = 3IT + 3FE = 3(23) + 3(88) = 333

<u>GOOP Algorithm</u>

MD = 2IT + 4FE = 2(28) + 4(115) = 516
AS = 2IT + 3FE = 2(28) + 3(115) = 401

Problem 2

a) Evaluation of function

MD = 5
AS = 3

b) The gradient is

$$\vec{g} = \begin{bmatrix} -600(x_2 - x_1^3)x_1^2 - 2(1 - x_1) \\ 200(x_2 - x_1^3) \end{bmatrix}$$

Evaluation of gradient

MD = 4
AS = 1

Total number of operations for function and gradient

MD = 9
AS = 4

c) The matrix H is

$$\begin{bmatrix} -30x_2^2 & 10 \\ -1 & 0 \end{bmatrix}$$

For construction of H

MD = 1
AS = 0

For H^TH and $H^T\vec{h}_0$

MD = 8
AS = 3

For scaling H^TH and $H^T\vec{h}_0$

MD = 3
AS = 0

Total number of operations for construction and scaling of H^TH and $H^T\vec{h}_0$

MD = 12
AS = 3

FR Algorithm

MD = 9FE = 9(44) = 395
AS = 4FE = 9(44) = 176

DFP Algorithm

MD = 9FE = 9(51) = 459
AS = 4FE = 4(51) = 204

LMC Algorithm

MD = 12IT + 5FE = 12(6) + 5(20) = 172
AS = 3IT + 3FE = 3(6) + 3(20) = 78

GOOP Algorithm

MD = 2IT + 5FE = 2(12) + 5(65) = 349
AS = 2IT + 3FE = 2(12) + 3(65) = 219

Problem 3

a) Evaluation of function

MD = 8
AS = 8

b) The gradient is

$$\vec{g} = \begin{bmatrix} -2(1.5 - x_1(1 - x_2))(1 - x_2) \\ \quad -2(2.25 - x_1(1 - x_2^2))(1 - x_2^2) \\ \qquad -2(2.625 - x_1(1 - x_2^3))(1 - x_2^3) \\ 2(1.5 - x_1(1 - x_2))x_1 \\ \quad + 4(2.25 - x_1(1 - x_2^2))\, x_1x_2 \\ \qquad + 6(2.625 - x_1(1 - x_2^3))x_1x_2^2 \end{bmatrix}$$

Evaluation of gradient

MD = 12
AS = 4

Total number of operations for function and gradient

MD = 20
AS = 12

c) The matrix H is

$$\begin{bmatrix} -1 + x_2 & x_1 \\ -1 + x_2^2 & 2x_1x_2 \\ -1 + x_2^3 & 3x_1x_2^2 \end{bmatrix}$$

For construction of H

MD = 4
AS = 0

For H^TH and $H^T\vec{h}_0$

MD = 12
AS = 8

For scaling H^TH and $H^T\vec{h}_0$

MD = 3
AS = 0

Total number of operations for construction and scaling of H^TH and $H^T\vec{h}_0$

MD = 19
AS = 8

<u>FR Algorithm</u>

MD = 20FE = 20(22) = 440
AS = 12FE = 12(22) = 264

<u>DFP Algorithm</u>

MD = 20FE = 20(24) = 480
AS = 12FE = 12(24) = 288

<u>LMC Algorithm</u>

MD = 19IT + 8FE = 19(5) + 8(15) = 215
AS = 8IT + 8FE = 8(5) + 8(15) = 160

<u>GOOP Algorithm</u>

MD = 3IT + 8FE = 3(10) + 8(50) = 430
AS = 3IT + 8FE = 3(10) + 8(50) = 430

Problem 4

a) Evaluation of function

MD = 10
AS = 7

b) The gradient is

$$\vec{g} = \begin{bmatrix} 2(x_1 + 10x_2) + 40(x_1 - x_4)^3 \\ 20(x_1 + 10x_2) + 4(x_2 - 2x_3)^3 \\ 10(x_3 - x_4) - 8(x_2 - 2x_3)^3 \\ -10(x_3 - x_4) - 40(x_1 - x_4)^3 \end{bmatrix}$$

Evaluation of gradient

MD = 8
AS = 4

Total number of operations for function and gradient

MD = 18
AS = 11

c) The matrix H is

$$\begin{bmatrix} 1 & 10 & 0 & 0 \\ 0 & 0 & \sqrt{5} & \sqrt{5} \\ 0 & 2(x_2 - 2x_3) & -4(x_2 - 2x_3) & 0 \\ 2\sqrt{10}(x_1 - x_4) & 0 & 0 & -2\sqrt{10}(x_1 - x_4) \end{bmatrix}$$

For construction of H

MD = 2
AS = 0

For H^TH and $H^T\vec{h}_0$

MD = 44
AS = 33

For scaling H^TH and $H^T\vec{h}_0$

MD = 16
AS = 0

Total number of operations for construction and scaling of H^TH and $H^T\vec{h}_0$

```
MD = 62
AS = 33
```

FR Algorithm

```
MD = 18FE = 18(123) = 2214
AS = 11FE = 11(123) = 1353
```

DFP Algorithm

```
MD = 18FE = 18(100) = 1800
AS = 11FE = 11(100) = 1100
```

LMC Algorithm

```
MD = 62IT + 10FE = 62(15) + 10(57) = 1500
AS = 33IT +  7FE = 33(15) +  7(57) =  894
```

GOOP Algorithm

```
MD =  4IT + 10FE = 4(16) + 10(120) = 1264
AS =  4IT +  7FE = 4(16) +  7(120) =  904
```

Problem 5

Problem 5 has the same function, gradient, and H matrix as problem 4. Because the starting points differ the number of function evaluations, FE, and the number of iterations, IT, also differ.

FR Algorithm

```
MD = 18FE = 18(121) = 2178
AS = 11FE = 11(121) = 1331
```

DFP Algorithm

```
MD = 18FE = 18(96) = 1728
AS = 11FE = 11(96) = 1056
```

LMC Algorithm

MD = 62IT + 10FE = 62(9) + 10(27) = 828
AS = 33IT + 7FE = 33(9) + 7(27) = 486

GOOP Algorithm

MD = 4IT + 10FE = 4(14) + 10(91) = 966
AS = 4IT + 7FE = 4(14) + 7(91) = 693

Problem 6

Problem 6 is the same as problem 4.

FR Algorithm

MD = 18FE = 18(57) = 1026
AS = 11FE = 11(57) = 627

DFP Algorithm

MD = 18FE = 18(54) = 972
AS = 11FE = 11(57) = 594

LMC Algorithm

MD = 62IT + 10FE = 62(6) + 10(18) = 552
AS = 33IT + 7FE = 33(6) + 7(18) = 324

GOOP Algorithm

MD = 4IT + 10FE = 4(8) + 10(61) = 642
AS = 4IT + 7FE = 4(8) + 7(61) = 459

Problem 7

a) Evaluation of function

MD = 13
AS = 13

b) The gradient is

$$\vec{g} = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) + 0.4(x_2 - 1) + \\ \quad 19.8(x_2 + x_4 - 2) \\ -360x_3(x_4 - x_3^2) - 2(1 - x_3) \\ 180(x_4 - x_3^2) + 0.4(x_4 - 1) + \\ \quad 19.8(x_2 + x_4 - 2) \end{bmatrix}$$

Evaluation of gradient

MD = 9
AS = 6

Total number of operations for function and gradient

MD = 22
AS = 19

c) The matrix H is

$$\begin{bmatrix} -20x_1 & 10 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -2\sqrt{90}x_3 & \sqrt{90} \\ 0 & -1 & 0 & 0 \\ \sqrt{0.2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{0.2} \\ 0 & \sqrt{9.9} & 0 & \sqrt{9.9} \end{bmatrix}$$

For construction of H

MD = 2
AS = 0

For H^TH and $H^T\vec{h}_0$

MD = 77
AS = 66

For scaling $H^T H$ and $H^T \vec{h}_0$

MD = 16
AS = 0

Total number of operations for construction and scaling of $H^T H$ and $H^T \vec{h}_0$

MD = 95
AS = 66

<u>FR Algorithm</u>

MD = 22FE = 22(61) = 1342
AS = 19FE = 19(61) = 1159

<u>DFP Algorithm</u>

MD = 22FE = 22(129) = 2838
AS = 19FE = 19(129) = 2451

<u>LMC Algorithm</u>

MD = 95IT + 13FE = 95(53) + 13(208) = 7739
AS = 66IT + 13FE = 66(53) + 13(208) = 6202

<u>GOOP Algorithm</u>

MD = 7IT + 13FE = 7(50) + 13(389) = 5407
AS = 7IT + 13FE = 7(50) + 13(389) = 5407

<u>Problem 8</u>

a) Evaluation of function

MD = 12
AS = 4
EXP = 1
TAN = 1

b) The gradient is

$$\vec{g} = \begin{bmatrix} 4(e^{x_1} - x_2)^3 e^{x_1} + 8x_1^{\,7} \\ -4(e^{x_1} - x_2)^3 + 600(x_2 - x_3)^5 \\ -600(x_2 - x_3)^5 + \dfrac{4\tan^3(x_3 - x_4)}{\cos^2(x_3 - x_4)} \\ \dfrac{-4\tan^3(x_3 - x_4)}{\cos^2(x_3 - x_4)} + 2(x_4 - 1) \end{bmatrix}$$

Evaluation of gradient

MD = 14
AS = 4
COS = 1

Total number of operations for function and gradient

MD = 26
AS = 8
EXP = 1
TAN = 1
COS = 1

c) The matrix H is

$$\begin{bmatrix} 2(e^{x_1} - x_2)e^{x_1} & 2(e^{x_1} - x_2) & 0 & 0 \\ 0 & 30(x_2 - x_3)^2 & -30(x_2 - x_3)^2 & 0 \\ 0 & 0 & \dfrac{2\tan(x_3 - x_4)}{\cos^2(x_3 - x_4)} & \dfrac{-2\tan(x_3 - x_4)}{\cos^2(x_3 - x_4)} \\ 4x_1^{\,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For construction of H

MD = 7
COS = 1

For $H^T H$ and $H^T \vec{h}_0$

MD = 55
AS = 44

For scaling $H^T H$ and $H^T \vec{h}_0$

MD = 16

Total number of operations for construction and scaling of $H^T H$ and $H^T \vec{h}_0$

MD = 78
AS = 44
COS = 1

<u>FR Algorithm</u>

MD =	26FE =	26(117) =	3042
AS =	8FE =	8(117) =	936
EXP =	1FE =	1(117) =	117
TAN =	1FE =	1(117) =	117
COS =	1FE =	1(117) =	117

<u>DFP Algorithm</u>

MD =	26FE =	26(180) =	4680
AS =	8FE =	8(180) =	1440
EXP =	1FE =	1(180) =	180
TAN =	1FE =	1(180) =	180
COS =	1FE =	1(180) =	180

<u>LMC Algorithm</u>

MD =	78IT +	12FE =	78(20) +	12(75) =	2460
AS =	44IT +	4FE =	44(20) +	4(75) =	1180
EXP =	0IT +	1FE =		1(75) =	75
TAN =	0IT +	1FE =		1(75) =	75
COS =	1IT +	0FE =	1(20)	=	20

<u>GOOP Algorithm</u>

MD =	5IT +	12FE =	5(10) +	12(80) =	1010
AS =	5IT +	4FE =	5(10) +	4(80) =	370
EXP =	0IT +	1FE =		1(80) =	80
TAN =	0IT +	1FE =		1(80) =	80
COS =	0IT +	0FE =			0

APPENDIX D

Stewart's Results for Problems 1 and 5*

TABLE 18

A Parabolic Valley

Iteration No.	Number of function evaluations	Φ
0	1	$2.4 \quad 10^{1}$
2	13	$3.8 \quad 10^{0}$
4	26	$2.9 \quad 10^{0}$
6	39	$1.9 \quad 10^{0}$
8	56	$7.1 \quad 10^{-1}$
10	70	$2.9 \quad 10^{-1}$
12	83	$1.4 \quad 10^{-1}$
14	97	$5.4 \quad 10^{-2}$
16	111	$1.8 \quad 10^{-2}$
18	124	$1.3 \quad 10^{-3}$
20	132	$1.7 \quad 10^{-6}$
22	145	$2.8 \quad 10^{-10}$
23	152	$1.0 \quad 10^{-11}$
24	163	$9.0 \quad 10^{-12}$
25	169	$3.3 \quad 10^{-12}$
26	174	$3.3 \quad 10^{-12}$

*These tables have been reproduced from Stewart's paper [9].

TABLE 19

A Function of Four Variables

Iteration No.	Number of function evaluations	Φ	
0	1	2.2	10^{2}
2	21	1.9	10^{1}
4	37	5.5	10^{-2}
6	54	1.1	10^{-2}
8	69	3.4	10^{-3}
10	83	2.7	10^{-3}
12	104	2.8	10^{-5}
14	122	1.8	10^{-7}
16	139	1.1	10^{-8}
18	158	8.8	10^{-9}
19	168	8.8	10^{-9}
.	.	.	.
.	.	.	.
.	.	.	.
41	407	1.1	10^{-10}